Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

Feb 14, 1995

Dear MacNosy/Debugger User,

Enclosed is the latest release of *The Debugger V2/95* and *MacNosy* and some accompanying notes on them.

This release is a complete replacement for any previous version.

For those of you who purchased *The Debugger* prior to November 1, 1994, this is you last free update. To receive updates and support in 1995, fill out the enclosed invoice and return it to me before May 1, 1995 with authorization to bill your credit card or a check for $130 (payable to a US Bank).
You can simplify my life by using e-mail to order your update subscription.

The 2/14/95 version of The Debugger is now the 'Reference version' or baseline for patches that I will be uploading to Internet and AppleLink. Send in your update fees so you get them.

Tech Support for The Debugger is available via phone, or e-mail (macnosy@jasik.com, ...).

Site (multiple copy) Licenses are available through Jasik Designs. Pricing is $225 per copy times the number of copies (programmers using the product) plus sales tax and $10 shipping.

General
Site Licensees - The CD-ROM contains a copy of this cover letter in Fullwrite format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

The CD-ROM contains the long awaited Debugger Tutorial. Finally !!

INIT notes & Other incompatibilities
xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
DiskLocker's (DiskLock, etc) appear to foul up the operation of *The Debugger*.
MicroSoft Mail must load **after** *The Debugger*.
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger*.
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger*.
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
INIT_ADB is an INIT to activate the 'interrupt' (Command-power) key on the IIsi and Mac LC machines. It is in the folder: '`Misc INITs & Source` '.
Retrospect Remote and *CCNotify* have been reported as incompatible with The Debugger.
Think Reference - Please use Version 2.0.1 or later
The Debugger is NOT compatible with Virtual Memory, RAM Doubler, ...

Installation Procedure
Copy the folder '`2/14/95 Debugger files` ' to your hard disk.
As a backup, it also contains a copy of the folder as a Self Extracting Archive (.sea) file.
If you have MPW, then I strongly suggest that you use the 'Install Debugger' script.
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
ROM .snt files are supplied for the PowerPC Macs's & Macs with '067C' ROMs (Mac IIci/IIfx, ...).
For other Macs will you can copy one from the folder 'more ROM .snt files' into the 'Dbgr/Nosy files' folder.
Boot the Mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

System Compatibility
The Debugger and *MacNosy* have been tested on Systems 6.0.7, 7.0 , 7.1, 7.1.2 and 7.5.

Miscellaneous Notes
• You may now **change command key equivalents** in The Debugger via the '`=Menu` ' command in the .dsi file. See the documentation in ROM.dsi for details. Now you can close windows with Cmd-W !!

•• You may need to use Set_Dbgr_Dir if you are playing with System 7.5

New Features and Fixes, etc. (see the "Change History" file for a complete list)

The file 'Change Summary - 2/14/95' contains a summary of changes since the last update.
The "'Change History" file contains a complete list of changes in chronological order.

This version of the product is compatible with the upcoming PCI Bus Macs.
You will have to use Nosy to build a ROM .snt from scratch.

- Cmd_Key_Plus flag - If FrontWindow is Read_only then 'Cmd key' is added to any Key press except 'e' and period. i.e. you can step by pressing the comma key when a code window is frontmost.

Fixed the PPC Watchpoint command so it is reasonable reliable.

- add ?stack(fba,stk_top,is_PPC) command to dump an arbitrary stack
  if stk_top = 0 then stk_top := fba + 0C00,
  is_PPC = 1 for stack that starts with PPC stack frames.

MacApp 3.1 Object Inspector - Apple has included my mods into MacApp 3.1 and later versions.

Nosy does not know about PEF (Power Executable Format) libraries, but YADP does.

PPCTraceEnabler goes in Extensions on Power Macs Only.
It is a bug fix version of the ProcessManagerSuport Library supplied by Apple.

- Source level debugging of ASLM (Apple Shared Library Mgr) V1.1 libraries is supported.
  The INST1_Bkpt flag has no effect, but you may set breakpoints in a .dsi file.
  You can debug Multi-seg Libraries. If the resource type to be debugged is NOT = 'code',
  then SYM file name must be of the form 'LibraryName.rrrr.SYM' where rrrr is the Resource type.
  ASLM Tasks are deleted when all segments are unloaded.
- add DBG_SLM flag so you can conveniently turn debugging of ASLM code ON/OFF

Known Problems

- The text drawing routines put garbage on the user's screen when using Step Continuous.
- When doing a Shutdown on some PowerBook 170's one gets a spurious BusError in *The Debugger*.
- Using MMU protection with programs built with Model FAR causes spurious errors, see DTN #7.
- Using the Think C Debugger in combination with *The Debugger* causes programs to stop at the end of The Debugger's patch to _LoadSeg as the TC Debugger sets LoadTrap (byte at $12D).
- ***The Debugger* does not work with System 7 VM**
- Do not try to use a Syquest with the R45 INIT as a boot disk, *The Debugger* will screw up.

Future Directions

At present, Nosy needs a bit of work to it so that it knows about the PowerPC and Code Fragments, etc. Another thing I have been thinking about is to try and 'hide' Nosy (or its function) so that The Debugger executes it on an as necessary basis when there is a crash or unresolved names in the stack crawl window, etc. I hope this will solve the problem of The Debugger not recognizing Macsbug names, etc.

There are many features that are available for 68K debugging that are still not available for PowerPC debugging. Memory Watch for the PowerPC works and will be extended so that it is like MMU Protection on the 030 Macs. Another area which I expect to pay more attention to this year is error checking, which includes extending Discipline to the PowerPC Library calls.

OpenDoc and SOM are coming, and I have to do something about displaying SOM Objects after Apple comes up with an API for it.

Shows I will Attend - I will be at the Apple WWDC in May.

If you did not receive notice of this mailing via an e-mail message,
then I do NOT have your e-mail address.
Please send it to me so I can check/update my database.

Sincerely,

Steve Jasik **Internet: macnosy@jasik.com Compuserve ID: 76004,2067 AppleLink: D1037**

10/12/95 Jasik Debugger Notes - October 12, 1995

To all Jasik Debugger users:

** The files in this folder/disk are -replacement- versions of 'The Debugger' and NosyII, ...
The files in this folder have NOT been tested extensively, and as such they -MAY- introduce bugs or break existing features.

** To take advantage of the new features, please copy -ALL- the files in the folder ' to Dbgr/Nosy files (10/12)' into the folder 'Dbgr/Nosy files' on your Hard Disk.

You may wish to backup your ROM.dsi file as I introduced a new flag:
'Load_Sys_Lib_Sym_Info'
Which controls the automatic loading of PowerPC Symbol Info for System Libraries.

*** General ***

*** Object Inspector for PowerPlant V1.2 ***
Is available in this archive AND on my WEB site.

*** Debugger Manual in Adobe Acobat Format ***
Is available in this archive AND on my WEB site.

*** 7500/8500/9500 bug alert ***
Apple put a nasty regression the 7.5.2(1.1) System that ships with the above Macs. The failure mode is that the SP register (r1 on the PPC) is incorrect after stepping over an instruction in my Debugger that changes it.
A few of you have stumbled upon it.
I put some special code in my Debugger to search for this bug and change the offending instruction in the System Library to the correct one.
Apple will fix this bug in some future release of the system later this fall.

*** System 7.5 upd 2.0 - s7 bug alert ***
They blew it on Sounds at INIT time again, and you will have to use 'Set_Dbgr_Directory' to point xDbgr_Startup to the location of 'The Debugger' OR you can just wait for 's8'.

*** Speeding up MPW Builds ***

Turn Filesharing OFF on your build Machine - 22 to 28 %
OR Kill the 'Finder'

Turn the Spinning Cursor OFF via the Shell Variable - 5 to 10%:

```
SET screenUpdateDelay 0
```

you can turn it on by: SET screenUpdateDelay 1

Compile multiple files via -ONE- compile directive:

```
MrC  fileA.c  fileB.c ... fileZ.c options ...
```

You can force 'make' to do this by playing with options.

** e-mail

e-mail works about 98% of the time, not 100% , -SO-:
Please include your name, address, Phone number
and business affiliation in your message
in the case that -MY- attempts to reply to you don't get through.

Please DO NOT send me enclosures with the name 'to Steve'
My download folder might fill up with Dup file names.
Instead, how about 'from yourName' ??

** Connectix RAM Doubler & my Debugger DO NOT work together.
De-install it and 're-bless' your System folder.

** Connectix Speed Doubler & my Debugger work together.
I am using it on my Power Macs and recommend it.

Please note that with 'Speed Doubler' installed:

a) You will not be able to use 'Performance Timing'

b) If your program is 'lost' in an infinite loop in 68K code
and you press Cmd-Power to enter any Debugger,
then you may end up at an 'FE10' instruction generated
by 'Speed Doubler' and will -NOT- be able to get
any information about where your program is.
The only solution is to re-run the test case with-out
Speed Doubler installed.

**** I have had various problem reports from some of you, and in
some cases the problems were traced to BUGGY Disk Drivers.
SilverLining 5.6.x 'klobbers' location 8 (the 68K bus Error vector),
which is totally un-acceptable.
The APS Drivers do wierd things at INIT time.
I recomend that you switch over to the FWB Drivers, and
purchase a copy of the FWB Hard Disk TookKit so as to avoid

such problems in the future (which are difficult to diagnose).

** Recommended versions of the Mac OS:

9500 - 7.5.2(1.1) Avail from Apple call 800-769-2775 x5617
I strongly suggest that you update to '1.1' version of the system.

8500 & 7500 - The System that comes with it (7.5.2(1.1)) is fine.

All Other Macs - we have been running '7.5 upd2.0-s6' as
shipped to developers on the 'Sept Developer Seed' CD-ROM
and it appears to be stable and eminately usable.

*** 9500/8500/7500 Alert ***

In general, this version of The Debugger is compatable with the new PCI Bus Macintoshs. Changes have been made to the cursor handling and the Watchpoint commands, etc.
BUT:

1) You need to make a new ROM/CODE.snt for these Macs from 'scratch'
Please use the latest version of NosyII (6/19/95 or later) to do it.

2) Using Cmd-Power to enter the Debugger will hang the machine
if you are using the 1.0 version of System 7.5.2 that was
shipped with the 1st batch of 9500's.
To avoid this problem, updgrade to '7.5.2(1.1)'

3) PCI Bus Macs & 2 monitors appears to work in ALL cases, but
Split screen may work best if in 'Monitors' you configure the
the boot screen and Menu Bar screen to be DIFFERENT monitors

** If you are getting the message 'BufPtr < SP' (from xDbgr_Startup) with System 7.5.x then you can try patching the 'boot',3 resource with program 'boot_rsrc_Patcher' which will automatically apply a patch to the 'System 7.5 Update' file or 'System' file as necessary so 'The Debugger' has more space available at Boot time.

To fix the problem in 7.5.x:
'Option copy' the file 'System 7.5 Update' to the Desktop and re-boot
Run 'boot_rsrc_Patcher' and select 'System 7.5 Update' (closed copy)
Move it back into the System folder.
Re-Boot

** Notes on the 'Low Memory Changed' message

The DebuggerINIT supplied by Metrowerks (> 6/10/95) and Jasik get along.
DebuggerINIT changes locations 18, 1C, 20, 24, ... & 90
Do NOT repair them.

When you flip the triangle in a MW project window, Apple's
Native QuickDraw 'Klobbers' location $10
The bug may be fixed in the next release (7.5 upd x.0 ??)

The SilverLining 5.6.x Disk driver 'Klobbers' location 8,
please replace it with 'FWB' Disk Drivers.

** Miscellaneous features of Interest:

* Files:'Create Log File' command for QA Testers, etc to save program state in a window which you can save to a file.
  Try it and you will see what it does.

* Add 'View As' as a Popup-Menu when 'Cmd-Option-MouseDown' (Π-click) OR as a click on the 'Triangle' in the Title of 'Type@address' windows.

  User may configure the types in the PopUp menu via the .dsi command:
  =X followed by one or more lines with typeNames (seperated by Commas)

  TO use the 'View As' command you may:

  0) Click in the 'triangle' in the Title bar of a 'Type@address'window.

  1) Hilight a 'Type@address' expression (hex address, etc) in ANY window (except code) and 'Π-click' to get a PopUp Menu of choices to 'cast' the expression. It will open up a new window displaying the data in the format you selected.
  Try it.
  2) If the cursor is in the Title bar of a 'type_@nnn' or unstructured asm window etc, then 'Π-click' to get a PopUp Menu of choices and it will reformat that window. When changing from TTT@nnn to @nnn, the code will remember TTT.

  3) If you click on a Name in a code window you will get a pop menu with the following: Procedure or Variable .
  This was done for those of you who complained that they could never remember which command to use (cmd-D , Cmd-W, Cmd-space, ...)

- -

***** Fixes/Changes 10/9/95 to 10/12/95 *****

Dbgr
fix - problems with Cursor on PCI Macs with 2 Monitors
fix - don't send the AppleEvent to run Nosy if Load_Sys_Lib_Sym_Info if OFF in ROM.dsi
fix - remove System patches are different, ... message as it was confusing

Nosy
fix - Alert user if more than 255 CODE segs, and only process 1st 255.
fix - misc bugs in disasm of Pef - 'QD GX' & 'WorldScript Power É'
fix - explore all procs that have TVectors
fix - In Global_Data, use prefixs: pV_nn = Ptr to Variable, pTV = Ptr to TVector
fix - in PEF if proc Len > 32K then split the proc up
fix - 'Nosy Status' dialog so it displays the Object we are processing

***** Fixes/Changes 9/13/95 to 10/9/95 *****

xDbgr_Startup
fix - swap scrn is default, do NOT lose split screen setting

Both
fix - treat 'type Ary_Name[]' from MW as 'Ary_Name:ARRAY[0:100] OF type'
fix - list mXspr instructions as mf/mt'spr' ri
* 'Set Source Path' - add a GetFile Dialog to interactively select it.
fix - when saving a ReadOnly window, clear the ReadOnly status

Dbgr
* Procedural Disassembly of PPC System Libraries is enabled by the flag: Load_Sys_Lib_Sym_Info.
Auto Load PPC System Library .snt files when there is a reference to an address in a 'System Context' CFM Library or the user does a Cmd-D on an address in one of the libraries.
* Display "*CurMap*" in the '-Task and File Info' window.
* Triangle where Zoom box is activates the View-As PopUp Menu
* Speed up Screen Swap on PPC by going native
* Cmd-D on 'nnn', 'c@nnn', É - display as PPC if in a CFM lib
allow Cmd-D on 'rd@nnn' & display proc if a single RD

fix - display of Trap calls on Exit when Bkpt at trap call
fix - suppress 'can't find source path' msg for CodeWarrior Lib path
fix - option-\ didn't get back to user when native _GNE call

fix - Apple Regression in 7.5.2(1.1) Apple ProcessMgrSupport 'sfvr'

failure was incorrect Stack Ptr after stepping in PPC mode.
Install code in Dbgr to fix it.
fix - update '-Dbgr Status-' window after an '=G' command
fix - revise CFM Notify Proc handling, allow main Appl with Exports
fix - .dsi command processing - save selection prior to executing
so an undo will erase any error and reselect the text
fix - '*p@(expr)' blew up if expr was an un-mapped address
fix - add warning that Modern Mem Mgr should be -ON-.
fix - Π-G displayed wrong zone name when MMM & Process Mgr's zone
fix - Handle typed as '^Ptr' caused 2 opt-spaces & field could not be modified
fix - PPC stepInst emulator - didn't setup FPSCR - got rounding errs on 8100/100

Nosy
* make creating a ROM/CODE.snt file less painful, especially for PCI Macs
* Disassemble PEF containers via selecting 'cfrg', DataFork, or
resorces containing PEF code fragments.
* Accept AppleEvents, Auto-Build PPC System Info when requested to
by 'The Debugger' - '.snt' files are saved in 'System Info Ä' .

***** Fixes/Changes 8/6/95 to 9/13/95 *****

Both
* process page up/down, home and End keys on the extended Keyboard
* PPC disasm: addic x,y,0 -> mr x,y & show ASCII const's as comments
fix - annotate trap calls where selector is CLR -(SP)
fix - minor revisions to asm listing format to make Nosy/PEF more readable

Nosy
fix - 'Cmd-Q' quits Nosy from the 4 button exit dialog
* disasm PPC CFM code fragments in data fork or in a resource, ...
* shift 'Display:Globals Map' displays the 'Global_data' declarations
and Cmd-D on 'Global_data' displays the block
* Add a fancy rsrc selection dialog
fix - name substitution logic in DisCmd.p
fix - dead carets in non-Front windows in name replace command (Cmd-h)
fix - .snt file, if no refs then clear rom_ref & sys_ref tbls

--Nosy Notes--
Nosy now disassembles 'PEF containers' with PowerPC code in them.
It does NOT work with for 'cfrg's with multiple containers (an odd case)
The Debugger' will read the '.snt' file of a 'PEF' object
You can also target the resultant '.snt' files in 'The Debugger'.
For the PEF stuff, this is an 'alpha' release as it is NOT feature complete.

```
Dbgr
  * identify this as Ver 2.8.3
  * enhance 'Target Lib' command so can read '.snt files' produced by Nosy PEF disasm
   fix - StepInto MixedModeMagic - didn't work for stack based selectors
   fix - Go Until PPC was broken
   fix - shift cmd-opt-Z ASCII = $FC , make Cmd key equiv work
   fix - Native Watchpoint was broken, it now works again
   fix - remove restriction that a CFM LIb must export a symbol to be debugged
   fix - code to use KCHR refed A5, incorrect as called at Interupt time
   fix - GrowZone Proc to release space from BufHndl if possible
   fix - cursor init logic so Screen swap might work on 2 Monitor PCI Bus Macs
         DisplayMgr is brain damaged, won't let me clone a GDevice
   fix - handling of -Values- so var with 0 offset works
   fix - revise Klobbered msg to be a bit clearer.
   fix - incorrect size calc for subranges with Upper Lim > 127
   fix - Perf timing wasn't working on the PCI bus Macs
         Note: - Perf Timing & Speed Doubler don't get along
   fix - remove extra screen swaps when processing a .sym file

xDbgr_Startup
   fix - swap scrn is default & accept Escape key to cancel
   fix - problems with APS drivers - add extra idiot msg

*****  Fixes/Changes  7/24/95  to  8/6/95  *****

xDbgr_Startup
   fix - check that name is: 'The Debugger' & is on Boot Partition
   fix -  improve NO space error msg & suggest using 'boot_rsrc_Patcher' to fix.
boot_rsrc_Patcher
  * new program to patch the boot,3 resource of System 7.5 & later

Dbgr
  * identify this as Ver 2.8.2
   fix - incorrectly setting is Case Sensitive flag for Pascal programs
   fix - option-\ caused a BusErr on the 9500 picked up wrong _WNE addr
   fix - make -Local Values- always come up as Dynamic
   fix - find_Hblk - no error checking for addrs in ROM Rsrc map sub-heap
   fix - modify getSuperClassID so 'class tree' display works for MacApp 3.3.1
   fix - screwup in System 7.5-s2 that caused _GetResource(MBDF) to fail
         when Speed Doubler was in.
  * .dsi files - =Open allow:  fileName,CLOSE to close an Opened .dsi window
   fix - no check for Mod Mem Mgr Dialog in System 7.5.2 & later
Nosy
```

fix - disassemble Metrowerks programs properly - still bugs in this area
fix - recognize data blks that start with LINK A6 inst and iteratively
explore them

Both
fix - AutoScrolling when Mouse is dragged outside the ViewRect

* show the names of the selectors for a wider variety of traps in the
assembly, -'Stack State-' and the '-Trap/User Call-' windows.
Cmd-D on '_Selector_trap_Name' will display the code of the underlying trap

***** Fixes/Changes 6/19/95 - 7/24/95 *****

Dbgr
fix - modify getSuperClassID so 'class tree' display works for MacApp 3.3.1
fix - INIT processing - let _CloseResFile delete the task as necessary
fix - _CloseResFile - don't blowup when Rsrc is Purged, just delete it
fix - could NOT modify fields with NIL or bad ptrs/handles
fix - rescrew invalid address checks - use Bus Error test for validity check
The flag SLOT_MEM is no longer refed and is always valid.
fix - deflect PEntryFromPSN (_OSDispatch) calls (PowerTalk Mail Server)
fix - PPC entry didn't check Low Mem if PC was Invalid
fix - incorrect identification of subheaps in Heap displays
fix - screwup in PPC disasm of abs addrs with bl $-$123456
fix - improve names in Stk crawl -
if ON PPC & addr in Sys heap, then look for CFM Lib name
display nearest exported proc name or ' $4000, '.sym.snt' was getting a spurious
read SST err =6
* NO more MMM Memorial Dialog Box on PCI Macs as they don't use the Exception
Handler
fix - add new selector for CoverTest
* if 1st char of a DebugStr is 'À' (opt-?) then write Log file to
'System folder' & resume execution
fix - Go Until so that it works over a Native call on PowerMacs
* display the ASCII equivalent of Longint values in'-Values-' windows
and Hex equiv of Integer, add sanity check to arrays with variable Upper Bounds
* When can't find a source path, copy it to '-Notes-' window
* Enable Jump Tracing for PowerPC Macs, use the PPC emulator to
collect the info. INcrease n_frames to 6 for OpenDoc/AppleScript
fix - Dbgr didn't get along with head patch to _SetCursor in 7.5.2-s2
fix - Reassign flags to they most frequent ones are 1st in list
Add Show_Alt_Fmt to control display of ASCII equiv of HLongint, ...
CoverText
* Add sanity error check to see if Len(blocks) in xSYM = Len Execuable Seg

Feb 96 Jasik CD ROM Release Notes

Feb 26, 1996

Dear MacNosy/Debugger User,

Enclosed is the latest release of *The Debugger V2/96* and *MacNosy* and some notes on them.

This release is a complete replacement for any previous version.

For those of you who purchased *The Debugger* prior to November 1, 1995, this is you last free update. To receive updates and support in 1996, fill out the enclosed invoice and return it to me before May 1, 1996 with authorization to bill your credit card or a check for $130 (payable to a US Bank).
You can simplify my life by ordering your Debugger update subscriptions via e-mail.

Last year, I posted most of my updates on the Internet and will continue to do so this year.
i.e. I should have an (internet) e-mail address for you on file.

The **2/24/96 version** of The Debugger is now the **'Reference version' or baseline for patches** that I will be uploading to Internet. **Send in your update fees so you get them.**

Tech Support for The Debugger is available via phone, or e-mail (macnosy@jasik.com).

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.
Site Licensees - This letter did not make it onto the CD-ROM, but is available on my WEB site.

Future Directions

For me the big push this year will be to get my Debugger running on Copeland.
You can help by sending in your update subscription fees ASAP.

New or Changed

In addition to the Hypercard *Debugger Tutorial*, I have added a *New Features '96' Tutorial.*

Exclusive - The CD-ROM contains a copy of **"The PowerPC™ Compiler Writers Guide"** in HTML format courtesy of IBM. For those of you who are not compiler writers, chapter 5 contains some neat coding tricks. I and a number of Macintosh developers/Apple employees contributed to the book. It will be available in bookstores later this year. **An errata sheet** for the manual is on IBM's web page.

Changed - I moved all the documentation over to *Adobe Acrobat*™ format. HTML versions will be available later this spring. The reference section of *The Debugger* manual was revised to bring it up to date.

Feature Changes in 95

1995 was a year of many changes for Nosy/The Debugger, compatibility with the PCI Macs headed the list.
A complete list of the feature changes is in the file: 'Feature Changes - 95' .
Some of the more notable changes are:

Nosy:
  New front end resource selection dialog making it easier to use.
  Disassembles PowerPC code and CFM containers.
  Disassembles Metrowerks 68K applications (funny jump table).
  Creating a ROM/CODE.snt file is less painful

The Debugger
  Soft MMU (heap centric bounds checking) for PPC programs.
  Enable Jump Tracing for PowerPC Macs.
  Update Bkpt & mmU sections of '.dsi' file when a task terminates (not compatible with 'dsi_close').
  If missing or wrong 'ROM/CODE.snt' file, then delete it and call NosyII to make one.
  Automatically debug PowerPC resources/CFM containers (OpenDoc Libraries)
  Add 'Create Log File' command for QA Testers, etc to save program state
  Add 'View As' Command as a PopupMenu in type@addr windows % as Cmd-opt-click
  Enhance PPC disassembly to show structure offsets, ...
  Speed up Screen Swap on PPC by going native
  Auto Load PPC System Library .snt files when there is a reference to an address in a
    'System Context' CFM Library or one does a Cmd-D on an addr in one of the libraries.
  Display of SOM Objects for OpenDoc users

Feb 96 Jasik CD ROM Release Notes

Feb 26, 1996

Dear MacNosy/Debugger User,

Enclosed is the latest release of *The Debugger V2/96* and *MacNosy* and some notes on them.

This release is a complete replacement for any previous version.

For those of you who purchased *The Debugger* prior to November 1, 1995, this is you last free update. To receive updates and support in 1996, fill out the enclosed invoice and return it to me before May 1, 1996 with authorization to bill your credit card or a check for $130 (payable to a US Bank).
You can simplify my life by ordering your Debugger update subscriptions via e-mail.

Last year, I posted most of my updates on the Internet and will continue to do so this year.
i.e. I should have an (internet) e-mail address for you on file.

The **2/24/96 version** of The Debugger is now the **'Reference version' or baseline for patches** that I will be uploading to Internet. **Send in your update fees so you get them.**

Tech Support for The Debugger is available via phone, or e-mail (macnosy@jasik.com).

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.
Site Licensees - This letter did not make it onto the CD-ROM, but is available on my WEB site.

Future Directions

For me the big push this year will be to get my Debugger running on Copeland.
You can help by sending in your update subscription fees ASAP.

New or Changed

In addition to the Hypercard *Debugger Tutorial*, I have added a *New Features '96' Tutorial*.

Exclusive - The CD-ROM contains a copy of **"The PowerPC™ Compiler Writers Guide"** in HTML format courtesy of IBM. For those of you who are not compiler writers, chapter 5 contains some neat coding tricks. I and a number of Macintosh developers/Apple employees contributed to the book. It will be available in bookstores later this year. **An errata sheet** for the manual is on IBM's web page.

Changed - I moved all the documentation over to *Adobe Acrobat*™ format. HTML versions will be available later this spring. The reference section of *The Debugger* manual was revised to bring it up to date.

Feature Changes in 95

1995 was a year of many changes for Nosy/The Debugger, compatibility with the PCI Macs headed the list.
A complete list of the feature changes is in the file: 'Feature Changes - 95' .
Some of the more notable changes are:
Nosy:
New front end resource selection dialog making it easier to use.
Disassembles PowerPC code and CFM containers.
Disassembles Metrowerks 68K applications (funny jump table).
Creating a ROM/CODE.snt file is less painful
The Debugger
Soft MMU (heap centric bounds checking) for PPC programs.
Enable Jump Tracing for PowerPC Macs.
Update Bkpt & mmU sections of '.dsi' file when a task terminates (not compatible with 'dsi_close').
If missing or wrong 'ROM/CODE.snt' file, then delete it and call NosyII to make one.
Automatically debug PowerPC resources/CFM containers (OpenDoc Libraries)
Add 'Create Log File' command for QA Testers, etc to save program state
Add 'View As' Command as a PopupMenu in type@addr windows % as Cmd-opt-click
Enhance PPC disassembly to show structure offsets, ...
Speed up Screen Swap on PPC by going native
Auto Load PPC System Library .snt files when there is a reference to an address in a 'System Context' CFM Library or one does a Cmd-D on an addr in one of the libraries.
Display of SOM Objects for OpenDoc users

Jasik Designs

343 Trenton Way   Menlo Park, CA 94025   (415) 322-1386

October 1, 1994

--Important Note--

I mail out updates to registered users every 3 to 5 months.
In between mailing I create patch files that apply to the
last mailed out version of The Debugger (9/30/94 in this case)
which include bug fixes and feature changes.

These updates are sent to registered users via e-mail
and in the case of new users placed on the distribution disks.

The 9/30/94 version of The Debugger 'escaped' with a few bugs in it,
and I strongly suggest that you install the enclosed patch
as per the below instructions.

The file 'mm_dd_Dbgr_Patch' updates 'The Debugger'.

**After you have unpacked the file: '9/30/94 Debugger .sea'
on your hard disk, you can install the patch:**

**A) Open up the folder 'Dbgr/Nosy files' and
change the name of 'The Debugger' to: '9/30/94 Debugger'**

B) Dbl-click on the UpdateMaker file 'mm_dd_Dbgr_Patch'

**C) Select the file '9/30/94 Debugger' and click in the OK box to create
a new version of 'The Debugger' (in the 'Dbgr/Nosy files' folder).**

**Note that the INIT 'xDbgr_Startup' which Launches 'The Debugger'
always looks for a file named 'The Debugger' .**

**You may verify that you are running the newer version
of The Debugger by checking the 4th line of the '-Notes-'
window when you re-boot with The Debugger installed.**

It should read:

```
USR_SG =    nnnn The Debugger - ... mm/dd/94
```

Notes:

On AppleLink, the path to my patch files is:

Developer Support: Developer talk: Debugger discussion:

On Internet:

ncftp ftp.netcom.com (use ftp if your system doesn't have ncftp)
cd /pub/macnosy

E-MAIL
Send me your INTERNET ADDRESS.

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

October 1, 1994

--Important Note--

I mail out updates to registered users every 3 to 5 months. In between mailing I create patch files that apply to the last mailed out version of The Debugger (9/30/94 in this case) which include bug fixes and feature changes.

These updates are sent to registered users via e-mail and in the case of new users placed on the distribution disks.

The 9/30/94 version of The Debugger 'escaped' with a few bugs in it, and I strongly suggest that you install the enclosed patch as per the below instructions.

The file 'mm_dd_Dbgr_Patch' updates 'The Debugger'.

After you have unpacked the file: '9/30/94 Debugger .sea' on your hard disk, you can install the patch:

A) Open up the folder 'Dbgr/Nosy files' and change the name of 'The Debugger' to: '9/30/94 Debugger'

B) Dbl-click on the UpdateMaker file 'mm_dd_Dbgr_Patch'

C) Select the file '9/30/94 Debugger' and click in the OK box to create a new version of 'The Debugger' (in the 'Dbgr/Nosy files' folder).

Note that the INIT 'xDbgr_Startup' which Launches 'The Debugger' always looks for a file named 'The Debugger' .

You may verify that you are running the newer version of The Debugger by checking the 4th line of the '-Notes-' window when you re-boot with The Debugger installed.

It should read:

```
USR_SG =    nnnn The Debugger - ... mm/dd/94
```

Notes:

On AppleLink, the path to my patch files is:

Developer Support: Developer talk: Debugger discussion:

On Internet:

ncftp ftp.netcom.com (use ftp if your system doesn't have ncftp)
cd /pub/macnosy

e-MAIL Send me your INTERNET ADDRESS.

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

Oct 16, 1995

Dear MacNosy User,

This year I installed an ISDN (high speed) connection to the Internet and have been posting updates to 'The Debugger' and MacNosy to my WEB/FTP site (http://www.jasik.com). This has proved to be less of a hassle for me and has allowed me to post updates to my product on a monthly basis

You are getting this letter and the enclosed disk because:

I do **NOT** have a **valid** e-mail address for you in my database.

Please note that next year, I may refuse to renew your Debugger/Nosy subscription if you still do NOT have a valid e-mail address.

I know that the above may sound a bit harsh, but at present about 93% of my users have e-mail addresses, and Apple, MicroSoft, ... are moving to on-line services for support. Note that Apple will be terminating AppleLink this fall. I suggest you consider joining eWorld, AOL, CompuServe or in Japan, NiftyServe or such.

Now about the floppy disk:

The file 10_12_Notes' describes the changes, new features, ...

The folder 'to Dbgr/Nosy files (10/12)' contains replacements for NosyII, The Debugger, ...
Copy the **contents** of it to your 'Dbgr/Nosy files' folder.

I also included a copy of The Debugger manual in Adobe Acrobat format and some other goodies.

Happy Debugging,

Steve Jasik

Steve Jasik

e-mail: **macnosy@jasik.com**
WWW: http://www.jasik.com

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

March 1, 1995

--Important Note--

1) ••• Wonders of Hypercard •••

Those of you who run the Debugger Tutorial on monitors smaller than 1152 x 872 (20", 21") should note that Hypercrud will reposition the main window to the center of the screen.
As the position of the movie window, ... is determined from the position of the main window, this will cause the Movie window to be partially off screen.
You can minimize or avoid this problem by dragging the Main Hypercard window to the upper left corner of the monitor **before** you play any movies.

2) I mail out updates to registered users every 3 to 5 months.
In between mailing I create patch files that apply to the **reference** or base version of The Debugger (2/14/95 in this case) which includes bug fixes and new features.

These updates are sent to registered users via e-mail and in the case of new users placed on the distribution disks.
You can install the enclosed patch as per the below instructions.

The file 'mm_dd_Dbgr_Patch' updates 'The Debugger'.
After you have copied the folder: '2/14/95 Debugger files' to your hard disk, you can install the patch:

10/12/95 and later patchs available on my web site

As of August 95, the patch mechanism broke down as the size of the patch files approached the size as the 'stuffed' programs.
The floppy disk that comes with my Debugger now contains a folder named: `'to Dbgr/Nosy files (mm/dd)'` ,
After doing a basic install of 'The Debugger', copy the **contents** of that folder **into** the folder `'Dbgr/Nosy files '` and you have installed the latest update.

In The Debugger, the second line of the '-Notes-' window should be:

```
USR_SG =    nnnn The Debugger V2.8.6...              10/12/95
```

or a later date

Notes:

The path to my patch files on Internet is:

WWW Home Page & files: http://www.jasik.com

Use Netscape to download the latest Patches.

You may also use Fetch or Anarchie with the URL:
'ftp://www.jasik.com//sjasik/

Default format for Files at the FTP site is 'MacBinary II'

From: macnosy@jasik.com (Steve Jasik)
Subject: Oct 12th Jasik Update News

To all Jasik Debugger users,

I don't like to do mailings to the entire group more frequently
than once a month, but almost everything that could go wrong with
an e-mailing went wrong last Monday (Oct 9th).

1) I didn't use BCC (Blind Carbon Copy) to send the messages and
everyone with a ccMail gateway returned my message back as the
list of names was too long. :-(

2) Given my past sucess in uploading archives, I pushed the limit
to 1.5Meg and some of you could NOT download it successfully.

3) There were some minor bugs that a few of you discovered, so
I tweeked Nosy to remove a DebugStr my assistant left in,
and in the meantime, I got a fix from Apple for the problem
of the cursor not behaving correctly on 2 Monitor PCI Mac systems.

SO, here is the revised archive that has some fixes, is smaller,
and finally you can get it via FTP.

****

A number of you asked me to send you a disk or to e-mail you
the archive.
---Lets start anew with this archive.---
i.e. - I am deleting your letters, ...

If you can't download via Netscape, or using FTP, then send me
an e-mail message, and I will e-mail or snail mail it to you.
****

-------Now for the details--------

The password for the files is: 'free the PPC ROM'
in honor the the changes
introduced in this version

The 10/12 patchs are in 2 places:

1) you get them the 'old' way via Netscape (800k as an .sea.hqx file).

2) FTP to 'www.jasik.com' , directory = '/sjasik' and it is the only
file in the Directory (10_12_Jasik_Mods.sea) at 550K.

In Anarchie,
the URL is: 'ftp://www.jasik.com//sjasik/10_12_Jasik_Mods.sea'
The file is formatted as: 'MacBinary II'

IN both cases,
expand the archive,
open up the folder: ' to Dbgr/Nosy files (10/12)'
Select All,
and copy the files into YOUR 'Dbgr/Nosy files' folder.

The file '10_12_Notes' describes the changes, ...

Steve

PS: I tested out both the FTP and Netscape Access to the files.
ON my ISDN line, Anarchie downloaded the files at 72KB per second !

Last minute stuff:

Bad OJ Jokes:

Now that OJ is free, he is thinking about starting a Limosine service.
The motto would be: 'we get you there on time, with at least an hour to kill'

OJ's e-mail address: slash, slash, backslash, escape

Serious stuff:

The Newer Accelerator for the 8500 and 9500 is available.
I have been using it and running my 9500 at 164 Mhz for the
last 3 weeks without any ill effects.

A few days ago one of my users at Adobe Seattle sent me an e-mail
that my Debugger wasn't working on his 9500/120
but was doing fine on a neighbor's 9500.

We had some discussions, he opened up his machine,
reseated everything, and wrote me the following:

>>>>>>
Subject: OUCH!!!!

Well, I re-seated the cards (and Processor card) and it started working!
Surprise, surprise. I'm still sending a flame to Apple about this. That
connector has to be redesigned.
<<<<<<<

This is the first series of Macintosh's where this problem has come
up with any frequency.

Most annoying !!

MY WEB SITE

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

Oct 1, 1994

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of *The Debugger V2/94* and *MacNosy* and some accompanying notes on them.
This release is a complete replacement for any previous version.

The 9/30/94 version of The Debugger is now the **'Reference version' or baseline** for patches that I will be uploading to AppleLink and Internet.

Tech Support for is available via phone, Applelink (D1037), or Internet (macnosy@netcom.com).

Please do NOT send messages from Internet to my Applelink address as Apple charges 50 cents per message.

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.

General

Site Licensees - The floppy contains a stuffed copy of this cover letter in MS Word 4 format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

INIT notes & Other incompatibilities

xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
DPI on the Fly - move to it to Extensions and make an Alias to it for use as a Control Panel.
DiskLocker's (DiskLock, etc) appear to foul up the operation of *The Debugger.*
MicroSoft Mail must load **after** *The Debugger.*
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger.*
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger*.
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
INIT_ADB is an INIT to activate the 'interrupt' (Command-power) key on the IIsi and Mac LC machines. It is in the folder: '`Misc INITs & Source`'.
Retrospect Remote and *CCNotify* have been reported as incompatible with The Debugger.

Installation Procedure

This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
If you have MPW, then I strongly suggest that you use the 'Install Debugger' script.
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
ROM .snt files are supplied for the PowerPC Macs's & Macs with '067C' ROMs (Mac IIci/IIfx, ...).
For other Macs will you have to create one. You must do this for AV and PowerBook 5xx Macintoshes.

Boot the mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

System Compatibility

The Debugger and *MacNosy* have been tested on Systems 6.0.7, 7.1, 7.1.2 & 7.5.
Remember to rename QuickDraw GX so it loads AFTER The Debugger.

Miscellaneous Notes

• You may now **change command key equivalents** in The Debugger via the '`=Menu`' command in the .dsi file. See the documentation in ROM.dsi for details. Now you can close windows with Cmd-W !!

• Cmd-D - if a partial name (Class::mm) is typed in, then open up the procedure that contains that name OR list all proc names that 'match' in a window.
For example, if `Txyz` is a class, then entering "`txyz::`" into the Cmd-D box will list all its methods.

•• You may need to use the APPL 'Set_Dbgr_Directory' if you are using with System 7.5

Miscellany

The last mailing was over 6 months ago, and that is a bit long for both you and me. My apologies.

The CD-ROM Tutorial is coming along. I expect to mail it out in ~~Nov~~, no before Christmas. :-)
All kidding aside, I want to get it out just as much as most of you want to see it.
We now have 400 Meg of Hypercard, Premier, etc on hard disk and are putting the finishing touches on it, ...

I am considering distributing my product on CD-ROM and dropping most of the paper.
Such a version of The Debugger would contain an electronic copy of the debugger manual, etc, along with a demo copy of FullWrite 2.0 which it is written in.
Comments from you are appreciated.

Now that Newsweek has a page devoted to computer/internet related matters I would expect that most of you would have e-mail addresses. If I don't have your e-mail address please send it to me.
I do have your e-mail address if you got a notice of this mailing on Oct 3rd via e-mail.

Patch files will be placed in :

AppleLink: Developer Support: Developer talk: Debugger discussion:
Internet: ncftp ftp.netcom.com ; cd /pub/macnosy

Disk Contents

'9/30 Feature Summary' (TEXT file) - a summary of the major external changes to The Debugger.

'READ ME !! - 9/30/94' (TEXT file) - a list of things NOT to do when installing/using The Debugger.
Most of it discusses problems that I am having with Apple hardware and software that I haven't resolved.

Please read it carefully as I have spent many hours on the phone with you discussing these problems.

'ETO 15 & Jasik - Link V3.4d2' (TEXT file) - a short note for MPW users.
Note that an upcoming version of Metrowerks will support the '3.3' SYM file format.

PowerPC Notes (MS Word) - some short notes on PowerPC debugging with The Debugger.

System 7.5 Sucks More ! (TEXT) - My response to the T-shirt (System 7.5 Sucks Less) which an Apple programmer was wearing at the Apple WWDC 1994.

9/30/94 Debugger.sea - A Self Extracting Archive containing the new release of The Debugger & Nosy.
It contains the following files of interest:

Changes 3/16/94 to 9/30 - unedited notes of all the changes to The Debugger, ... in the last 6 months.

Misc Nosy Files - 9/94 - Used to be called the 'Blue disk' An electronic copy of the Debugger Tech Notes are hiding inside it.

NEW Debugger Scripts - scripts to display thread Manager Queues, ...

Discipline Stuff - contains a Double Dispose Example - run it once to familiarize yourself with my error msg.

9/30/94 Debugger files - The 'Install Debugger (MPW)' script is hiding in here.
I could have moved it up a level, but that would take away from the fun of a good scavenger hunt.

New Human Interface Features - some comments

As I mentioned in the '9/30 Feature Summary' I am not completely happy with the implementation of the direct memory change in the different windows, I am also having second thoughts on the use of the Function Keys to implement one key debugger commands. Another possibility is to note that most of the windows in The Debugger are read-only and hence I could allow one to enter commands without having to press the command key when one of those windows is frontmost. It is trivial to implement and you can expect to see some variant of it in the near future.

PowerPC Notes - Sept 30th 1994

Learning about the PowerPC

I can display info about Power Macs, but I am NOT in the business of developing reference materials.
The PowerPC Macs have a very different architecture from the 68K Macs.
The runtime environment of the PowerPC Macs is derived from IBM's AIX OS.

To get started, buy the 'Macintosh with powerPC Starter Kit" from APDA (R0563LL/A) for about $40.00 .
This is essential. The kit contains the Motorola/IBM reference manual for the 601 CPU.

Also consider buying the 'Programmer's Introduction to RISC & PowerPC" a self paced mini-course for $150 (R0172LL/A).
If you or your company will spring for it, consider attending one of Apple's Developer University courses.

The Debugger vs Power Macintosh (as of 9/30/94)

The Debugger is capable of debugging Power Mac (Native mode) programs.
When you enter The Debugger, the format of the display in the registers window is dependent on the mode the machine was in when the exception was taken. If the machine was in Native mode, then you will get a display of the PowerPC registers, else you will get the classic 68K register display.

Features **not yet** implemented for native programs are:

Proc Entry/Exit Trace
Trap Discipline
Trap Intercept & Trap Entry Trace
Transparent debugging of Executable resources. (use 'Target Lib' as described below)
Intelligent step into for mixed mode calls.

Jump Tracing	may never be available for Native programs
MMU Protection	may never be available for Native programs

Structured disassembly of Native code in the ROMs. (Kludges are in Nosy for display in The Debugger)
Nosy doesn't know how to disassemble the PowerPC (CFM) libraries, and IBS hasn't been touched.

Some of these deficiencies will be rectified in the next few months,
along with getting the Debugger to work with VM.

Feature that are now working include:

Single statement source stepping.
Performance Timing
Memory Watch
CoverTest knows how to do Code coverage of Native programs,

Features added for PowerPC debugging

PowerPC disassembly with *'training wheels'* and emulation trace
The format of a line in a structured assembly window is:

```
   addr: result   label   opcode  operands     ; meaning of instruction
```

The result field may hold the following: the result value of a load or register to register instruction, or '>address' , the effective address of a store instruction, or 'except n' if execution of the instruction will cause an exception.

In the Tables menu, two new items have been added: 'CFM...' and 'Target Lib...' .

The CFM item will display a list of all the loaded CFM libraries, or of all the entry points in a given library, or it will find the address of a given symbol.

The 'Target Lib' command allows you to specify a library to be symbolically debugged.
You must have an 'xSYM' file for the library. To use the command, select it once, dismiss the dialog, hilight the name of the library you want to debug, and then select the '.xSYM' file from the Standard file dialog.

You may reference the pseudo variables: ?LR, ?CR, ?XER, ?CTR in expressions.

PowerPC Notes - Sept 30th 1994

Learning about the PowerPC

I can display info about Power Macs, but I am NOT in the business of developing reference materials.
The PowerPC Macs have a very different architecture from the 68K Macs.
The runtime environment of the PowerPC Macs is derived from IBM's AIX OS.

To get started, buy the 'Macintosh with powerPC Starter Kit" from APDA (R0563LL/A) for about $40.00 .
This is essential. The kit contains the Motorola/IBM reference manual for the 601 CPU.

Also consider buying the 'Programmer's Introduction to RISC & PowerPC" a self paced mini-course for $150 (R0172LL/A).
If you or your company will spring for it, consider attending one of Apple's Developer University courses.

The Debugger vs Power Macintosh (as of 9/30/94)

The Debugger is capable of debugging Power Mac (Native mode) programs.
When you enter The Debugger, the format of the display in the registers window is dependent on the mode the machine was in when the exception was taken. If the machine was in Native mode, then you will get a display of the PowerPC registers, else you will get the classic 68K register display.

Features **not yet** implemented for native programs are:

Proc Entry/Exit Trace
Trap Discipline
Trap Intercept & Trap Entry Trace
Transparent debugging of Executable resources. (use 'Target Lib' as described below)
Intelligent step into for mixed mode calls.

Jump Tracing	may never be available for Native programs
MMU Protection	may never be available for Native programs

Structured disassembly of Native code in the ROMs. (Kludges are in Nosy for display in The Debugger)
Nosy doesn't know how to disassemble the PowerPC (CFM) libraries, and IBS hasn't been touched.

Some of these deficiencies will be rectified in the next few months,
along with getting the Debugger to work with VM.

Feature that are now working include:

Single statement source stepping.
Performance Timing
Memory Watch
CoverTest knows how to do Code coverage of Native programs,

Features added for PowerPC debugging

PowerPC disassembly with *'training wheels'* and emulation trace
The format of a line in a structured assembly window is:

```
   addr: result   label   opcode  operands     ; meaning of instruction
```

The result field may hold the following: the result value of a load or register to register instruction, or '>address' , the effective address of a store instruction, or 'except n' if execution of the instruction will cause an exception.

In the Tables menu, two new items have been added: 'CFM...' and 'Target Lib...' .

The CFM item will display a list of all the loaded CFM libraries, or of all the entry points in a given library, or it will find the address of a given symbol.

The 'Target Lib' command allows you to specify a library to be symbolically debugged.
You must have an 'xSYM' file for the library. To use the command, select it once, dismiss the dialog, hilight the name of the library you want to debug, and then select the '.xSYM' file from the Standard file dialog.

You may reference the pseudo variables: ?LR, ?CR, ?XER, ?CTR in expressions.

Invoice

March 18, 1994

From: Jasik Designs, 343 Trenton Way, Menlo Park, CA 94025
(415) 322-1386

To: paste your mailing label here

New Address ______________________
____
____

For:

Quantity	Item	Unit price	Total
_____	**Debugger/Nosy Updates in 1994**	**$130**	_____
_____	Additional Licenses for Univ Debugger	**$225**	_____
	8.2 % Sales Tax for California Residents		_____
		Total	_____

If you are paying by Visa/MC then fill out:

Card Name: ___ **VISA** ___ **MasterCard**

Card Number: _____-_____-_____-_____ **Expires:** ___/___

Name of Cardholder (Print) ______________________________

Your Signature: ______________________________

Phone (in case of problems) ____________________ **AppleLink #** _________

Orders paid by credit card may be sent to me via AppleLInk (D1037) or CompuServe (76004,2067) , etc.
Please include your card number, expiration date, name, and authorization to debit your card by the stated amount.

Payment must accompany this form !!!
DO NOT send me a Purchase Order !!

Orders paid for by Visa/MasterCard will show up on your Visa/MC statement as a charge from **Jasik Designs**.
Paper receipts will be supplied **only upon request**.

You are being billed for update fees for your copies of *The Debugger V2 & Nosy* because my records show that you purchased/updated your copies of it prior to Dec. 1, 1993.

If you feel that I am in error, then please submit a copy of canceled checks or other proof of purchase.

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

March 17, 1994

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of *The Debugger V2/94* and *MacNosy* and some accompanying notes on them. **This release is (one) complete, and (two) a replacement for any previous version.**

For those of you who purchased *The Debugger* prior to December 1, 1993, this is you last free update. To receive updates and support in 1994, fill out the enclosed invoice and return it to me before June 1, 1994 with authorization to bill your credit card or a check for $130 (payable to a US Bank).
You can simplify my life by using e-mail to send in your update fees.
The 3/16/94 version of The Debugger is now the 'Reference version' for patches that I will be uploading to AppleLink and Internet. Send in your update fees so you get them.

Tech Support for The Debugger is available via phone, Applelink (D1037), or CompuServe (70277,776).

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.

General
Site Licensees - The floppy contains a stuffed copy of this cover letter in MS Word 4 format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

INIT notes & Other incompatibilities
xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
DPI on the Fly - move to it to Extensions and make an Alias to it for use as a Control Panel.
DiskLocker's (DiskLock, etc) appear to foul up the operation of *The Debugger.*
MicroSoft Mail must load **after** *The Debugger.*
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger.*
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger.*
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
INIT_ADB is an INIT to activate the 'interrupt' (Command-power) key on the IIsi and Mac LC machines. It is in the folder: '`Misc INITs & Source`'.
Retrospect R mote and *CCNotify* have been reported as incompatible with The Debugger.

Installation Procedure
This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
If you have MPW, then I strongly suggest that you use the 'Install Debugger' script.
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
Copy or create a ROM .snt for your machine (default in Dbgr/Nosy files is for a $067C ROM (Mac IIci/IIfx, ...)
You must do this for AV and Power Macintoshes.
Boot the mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

System Compatibility
The Debugger and *MacNosy* have been tested on Systems 6.0.5, 6.0.7, 7.0 , 7.0.1, 7.1, & 7.1.2.

Miscellaneous Notes
• You may now **change command key equivalents** in The Debugger via the '`=Menu`' command in the .dsi file. See the documentation in ROM.dsi for details. Now you can close windows with Cmd-W !!

• Cmd-D - if a partial name (Class::mm) is typed in, then open up the procedure that contains that name
OR list all proc names that 'match' in a window.
For example, if `Txyz` is a class, then entering "`txyz::`" into the Cmd-D box will list all its methods.

•• You may need to use Set_Dbgr_Dir if you are playing with System 7.5

PowePC Notes - March 17th 1994

Learning about the PowerPC

I can display info about Power Macs, but I am NOT in the business of developing reference materials.
The PowerPC Macs have a very different architecture from the 68K Macs.
The runtime environment of the PowerPC Macs is derived from IBM's AIX OS.

To get started, buy the 'Macintosh with powerPC Starter Kit" from APDA (R0563LL/A) for about $40.00 . **This is essential.** The kit contains the Motorola/IBM reference manual for the 601 CPU.

Also consider buying the 'Programmer's Introduction to RISC & PowerPC" a self paced mini-course for $150 (R0172LL/A).
If you or your company will spring for it, consider attending one of Apple's Developer University courses.

The Debugger vs Power Macintosh (as of 3/16/94)

The Debugger is capable of debugging Power Mac (Native mode) programs.
When you enter The Debugger, the format of the display in the registers window is dependent on the mode the machine was in when the exception was taken. If the machine was in Native mode, then you will get a display of the PowerPC registers, else you will get the classic 68K register display.

At present, not all features of the 68K debugger have been implemented in native mode.
Features **not** implemented for native programs include:

Single statement source stepping (it is at the instruction level). Using Go Until PC is a bit faster.
Performance Timing
Intelligent step into for mixed mode calls.
Memory Watch and Proc Entry/Exit Trace
Trap Discipline
Trap Intercept & Trap Entry Trace
Transparent debugging of Executable resources.
Structured disassembly of Native code in the ROMs.

Jump Tracing	may never be available for Native programs
MMU Protection	may never be available for Native programs

In addition, CoverTest does not know how to do Code coverage of Native programs,
Nosy doesn't know how to disassemble the PowerPC (CFM) libraries, and IBS hasn't been touched.

Hopefully many of these deficiencies will be rectified in the next few months,
along with getting the Debugger to work with VM.

Features added for PowerPC debugging

PowerPC disassembly with 'training wheels' and emulation trace
The format of a line in a structured assembly window is:

```
  addr: result   label   opcode  operands     ; meaning of instruction
```

The result field may hold the following: the result value of a load or register to register instruction, or '>address' , the effective address of a store instruction, or 'except n' if execution of the instruction will cause an exception.

In the Tables menu, two new items have been added: 'CFM...' and 'Target Lib...' .

The CFM item will display a list of all the loaded CFM libraries, or of all the entry points in a given library, or it will find the address of a given symbol.

The 'Target Lib' command allows you to specify a library to be symbolically debugged.
You must have an 'xSYM' file for the library. To use the command, select it once, dismiss the dialog, hilight the name of the library you want to debug, and then select the '.xSYM' file from the Standard file dialog.

You may reference the pseudo variables: ?LR, ?CR, ?XER, ?CTR in expressions.

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

December 30, 1993

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of the Universal Version of *The Debugger V2/93* and *MacNosy* and some accompanying notes on them. This release contains numerous bug fixes and new features.
Major feature enhancements included are MacApp 3.1ß1 support and PowerPC Native mode debugging.

Tech Support for The Debugger is available via phone, Applelink (D1037), CompuServe or Internet.

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.

Metrowerks Codewarrior is here, and unlike Symantec C, it produces an Apple standard .SYM file. You get FULL source level debugging with Metrowerks. **I strongly encourage you to give it a try.**

General
Site Licensees - The floppy contains a stuffed copy of this cover letter in MS Word 4 format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

I have includes a number of 'goodies' on the disk, including *Swatch* (a Heap Watcher by Joe Holt of Adobe), an electronic copy of gray manual in text format, and some Discipline examples.

INIT notes & Other incompatibilities
xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
DPI on the Fly - move to it to Extensions and make an Alias to it for use as a Control Panel.
DiskLocker's (DiskLock, etc) appear to foul up the operation ot *The Debugger.*
MicroSoft Mail must load **after** *The Debugger.*
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger.*
SuperBoomerang- use V3.01 or later as some of the 3.0 versions had bugs.
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger.*
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
Retrospect Remote and *CCNotify* have been reported as incompatible with The Debugger.
QuickKeys 3.0 appears to cause problems when Shutdown is selected from inside The Debugger.
AOCE AppleMail V1.0 - the disk contains a patch file to make it compatable with The Debugger.

Think C Reference V2 is incompatible with The Debugger, Use Version 2.0.1 which is compatible.

Installation Procedure
This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
The Debugger is now shipped on one 1.4 Meg floppy. This is a complete release.

If you have MPW, then I strongly suggest that you use the 'Install Debugger' script.
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
Copy or create a ROM .snt for your machine (default in Dbgr/Nosy files is for a $067C ROM (Mac IIci…)
Boot the mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

I will distribute Patch files to this version of The Debugger via AppleLink.
Make a copy of The Debugger (call it '12/30 Debugger') to run the patcher against.

System Compatibility
The Debugger and *MacNosy* have been tested on Systems 6.0.5, 6.0.7, 7.0, 7.0.1, 7.1, …
Fixes to *The Debugger* this spring make it compatible with WorldScript II, QuickDraw GX, AOCE, etc.

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

December 30, 1993

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of the Universal Version of *The Debugger V2/93* and *MacNosy* and some accompanying notes on them. This release contains numerous bug fixes and new features. **Major feature enhancements included are MacApp 3.1ß1 support and PowerPC Native mode debugging.**

Tech Support for The Debugger is available via phone, Applelink (D1037), CompuServe or Internet.

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.

Metrowerks Codewarrior is here, and unlike Symantec C, it produces an Apple standard .SYM file. You get FULL source level debugging with Metrowerks. **I strongly encourage you to give it a try.**

General
Site Licensees - The floppy contains a stuffed copy of this cover letter in MS Word 4 format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

I have includes a number of 'goodies' on the disk, including *Swatch* (a Heap Watcher by Joe Holt of Adobe), an electronic copy of gray manual in text format, and some Discipline examples.

INIT notes & Other incompatibilities
xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
DPI on the Fly - move to it to Extensions and make an Alias to it for use as a Control Panel.
DiskLocker's (DiskLock, etc) appear to foul up the operation of *The Debugger.*
MicroSoft Mail must load **after** *The Debugger.*
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger.*
SuperBoomerang- use V3.01 or later as some of the 3.0 versions had bugs.
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger.*
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
Retrospect Remote and *CCNotify* have been reported as incompatible with The Debugger.
QuickKeys 3.0 appears to cause problems when Shutdown is selected from inside The Debugger.
AOCE AppleMail V1.0 - the disk contains a patch file to make it compatable with The Debugger.

Think C Reference V2 is incompatible with The Debugger, Use Version 2.0.1 which is compatible.

Installation Procedure
This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
The Debugger is now shipped on one 1.4 Meg floppy. This is a complete release.

If you have MPW, then I strongly suggest that you use the 'Install Debugger' script .
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
Copy or create a ROM .snt for your machine (default in Dbgr/Nosy files is for a $067C ROM (Mac IIci…)
Boot the mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

I will distribute Patch files to this version of The Debugger via AppleLink.
Make a copy of The Debugger (call it '12/30 Debugger' **) to run the patcher against.**

System Compatibility
The Debugger and *MacNosy* have been tested on Systems 6.0.5, 6.0.7, 7.0, 7.0.1, 7.1, …
Fixes to *The Debugger* this spring make it compatible with WorldScript II, QuickDraw GX, AOCE, etc.

Debugger Tech Note #11

Parser Extensions & 'dcmds'

8/93, Revised 12/93

The basic expression language is described on pages 37 to 40 of the gray manual and summarized in the '-D_Help-' file which is displayed when the Help command is selected in The Debugger.
This tech note discusses additions to the expression language that are not in the gray manual.
They make it useful for the writing of scripts which display the contents of tables, lists, etc.

The AddMenu .dsi command (=Addmenu)

This .dsi file command lets you associate a Menu item in the "MacApp" Menu with a script. The Syntax is:

```
=Addmenu
'MenuItem_Name'/CmdKey
action clause lines
```

To delete an AddMenu from the "MacApp" Menu one hiltes ONLY the =A and addmenu name lines.
The file "Dbgr Scripts" contains some sample AddMenu scripts.

The ?get_value function

When the ?get_value function is executed from inside an AddMenu script it returns the hexidecimal value that the user entered, or was hilited in the front window. Its syntax is:

```
        Function  ?get_Value('prompt_string'):Longint;
An example of it's use is:   ?fba := ?get_Value('enter 1st byte addr to dump');
```

If no selection is present, then the user will be presented with a 2 line typein dialog box with the prompt string on the second line. If a selection is present, then it will be used as the value to be converted.
Leading garbage (@, $, =) in the string will be skipped over. The string is assume to be a hex number.
If the user presses the Cancel button, then the script is terminated.

Output Re-direction

Normally commands, write statements in 'action clauses', etc put their output at the end of the '-Notes-' window. The ?ReDirect command lets one define an alternate window for such output. The syntax is:

`?ReDirect('windowName' {,Append} );` - If 'windowName' exists then bring it to the front, else create a window with a name 'windowName' . If the Append parameter is present, then output will be appended to the end of the text in the window, else the text will be deleted and the window drawn as empty the next time it is written to.
`?ReDirect;` - re-directs output to the '-Notes-' window in Append mode.

Note: Unlike MPW, The Debugger always redirects output to the end of text in the redirection window.
Note: If the shift key is held down when an Addmenu is executed, then ?ReDirect('wName') will always create a new window with the name 'wName nn' (nn - 1, 2, ...).

Array References

You may reference the elements of a 1-dimensional array via the notation: `ArName[subScript_expr]` .
If ArName is a pointer to an Array, then one writes: `ArName^[subScript_expr]` and so on.
The address is calculated as:

    AddrOf(ArName) + sizeOf_Arname_element * (subScript_expr - Lower_bound)

which is just what you would expect. For an example of it's use see the 'Files' script in 'Dbgr Scripts'.
You can reference any of your global variables in your program during the execution of a script.
The way to display an element of an array of `Extended` elements is: `write(myArray[n]:Extended);`

Debugger Tech Note #11

Parser Extensions & 'dcmds'

8/93, Revised 12/93

The basic expression language is described on pages 37 to 40 of the gray manual and summarized in the '-D_Help-' file which is displayed when the Help command is selected in The Debugger.
This tech note discusses additions to the expression language that are not in the gray manual.
They make it useful for the writing of scripts which display the contents of tables, lists, etc.

The AddMenu .dsi command (=Addmenu)

This .dsi file command lets you associate a Menu item in the "MacApp" Menu with a script. The Syntax is:

```
=Addmenu
'MenuItem_Name'/CmdKey
action clause lines
```

To delete an AddMenu from the "MacApp" Menu one hiltes ONLY the =A and addmenu name lines.
The file "Dbgr Scripts" contains some sample AddMenu scripts.

The ?get_value function

When the ?get_value function is executed from inside an AddMenu script it returns the hexidecimal value that the user entered, or was hilited in the front window. Its syntax is:

```
        Function  ?get_Value('prompt_string'):Longint;
An example of it's use is:  ?fba := ?get_Value('enter 1st byte addr to dump');
```

If no selection is present, then the user will be presented with a 2 line typein dialog box with the prompt string on the second line. If a selection is present, then it will be used as the value to be converted.
Leading garbage (@, $, =) in the string will be skipped over. The string is assume to be a hex number.
If the user presses the Cancel button, then the script is terminated.

Output Re-direction

Normally commands, write statements in 'action clauses', etc put their output at the end of the '-Notes-' window. The ?ReDirect command lets one define an alternate window for such output. The syntax is:

`?ReDirect('windowName' {,Append} );` - If 'windowName' exists then bring it to the front, else create a window with a name 'windowName' . If the Append parameter is present, then output will be appended to the end of the text in the window, else the text will be deleted and the window drawn as empty the next time it is written to.
`?ReDirect;` - re-directs output to the '-Notes-' window in Append mode.

Note: Unlike MPW, The Debugger always redirects output to the end of text in the redirection window.
Note: If the shift key is held down when an Addmenu is executed, then ?ReDirect('wName') will always create a new window with the name 'wName nn' (nn - 1, 2, ...).

Array References

You may reference the elements of a 1-dimensional array via the notation: `ArName[subScript_expr]` .
If ArName is a pointer to an Array, then one writes: `ArName^[subScript_expr]` and so on.
The address is calculated as:

AddrOf(ArName) + sizeOf_Arname_element * (subScript_expr - Lower_bound)

which is just what you would expect. For an example of it's use see the 'Files' script in 'Dbgr Scripts'.
You can reference any of your global variables in your program during the execution of a script.
The way to display an element of an array of `Extended` elements is: `write(myArray[n]:Extended);`

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

August 2, 1993

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of the Universal Version of *The Debugger V2/93* and *MacNosy* and some accompanying notes on them. This release contains numerous bug fixes and new features.

Tech Support for The Debugger is available via phone, Applelink (D1037), or CompuServe.
I have also signed up to access Internet via 'Netcom', and will make appearances there later this summer after I figure out how to use the Unix (Yeech!) interface. My address is: `macnosy@netcom.netcom.com` .

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.

General

Site Licensees - The floppy contains a stuffed copy of this cover letter in MS Word 4 format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

This update is being shipped on a 1.4 Meg Floppy.
If you don't have access to a Mac IIx or later Mac then I can send you the update on 800K floppies.
I have includes a number of 'goodies' on the disk, including *Swatch* (a Heap Watcher by Joe Holt of Adobe), an electronic copy of gray manual in text format, and some Discipline examples.

Any updates to The Debugger over the next few months will be posted in the 'Debugger Discussion' area on AppleLink as patches to the 8/1/93 version of The Debugger.

INIT notes & Other incompatibilities

xDbgr_Startup MUST be placed in the Extensions folder on System 7.
DPI on the Fly - move to it to Extensions and make an Alias to it for use as a Control Panel.
DiskLocker's (DiskLock, etc) appear to foul up the operation of *The Debugger.*
MicroSoft Mail must load **after** *The Debugger.*
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger.*
SuperBoomerang- use V3.01 or later as some of the 3.0 versions had bugs.
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger.*
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
INIT_ADB is an INIT to activate the 'interrupt' (Command-power) key on the IIsi and Mac LC machines. It is in the folder: '`Misc INITs & Source`'.
Retrospect Remote and *CCNotify* have been reported as incompatible with The Debugger.

Think C Reference V2 is incompatible with The Debugger, Use Version 2.0.1 which is compatible.

Installation Procedure

This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
If you have MPW, then I strongly suggest that you use the 'Install Debugger' script .
To manually install *The Debugger,* do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
Copy or create a ROM .snt for your machine (default in Dbgr/Nosy files is for a $067C ROM (Mac IIci/IIfx, ...)
Boot the mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

System Compatibility

The Debugger and *MacNosy* have been tested on Systems 6.0.5, 6.0.7, 7.0, 7.0.1, 7.1, ...
Fixes to *The Debugger* this spring make it compatible with WorldScript II, QuickDraw GX, AOCE, etc.

Jasik Discipline vs Apple code

I fixed some spurious discipline errors in The Debugger only to find some fairly evil code emanating from Cupertino. Check out the 'Discipline Stuff' folder for a discussion of this and an interesting example.

MPW's *ILink* vs *The Debugger*

I will NOT be supporting the State (.NJ) files produced by ILink which is part of ETO #10.
I feel that IBS is still a superior solution, and urge you to give it a try if you haven't already.

New Features and Fixes, etc. (see the "Change History" file for a complete list)

The file 'Change Summary- 3/27 to 8/2/93' contains a summary of changes since the last update.
The "Change History" file contains a complete list of changes in chronological order.
DTN #11 describes the changes to the parser, which include support for Macsbug 'dcmds'.

CoverTest - now works with ThC projects, the documentation has been revised.
The Debugger will search the project for the names of Static functions,
and name them 'fileName$funcName' or 'fileName$static_n' is source is not available.

- Source level debugging of ASLM (Apple Shared Library Mgr) V1.1 libraries is supported.
  The INST1_Bkpt flag has no effect, but you may set breakpoints in a .dsi file.
  You can debug Multi-seg Libraries. If the resource type to be debugged is NOT = 'code',
  then SYM file name must be of the form 'LibraryName.rrrr.SYM' where rrrr is the Resource type.
  ASLM Tasks are deleted when all segments are unloaded.
- add DBG_SLM flag so you can conveniently turn debugging of ASLM code ON/OFF

Known Problems

- The text drawing routines put garbage on the user's screen when using Step Continuous.
- When doing a Shutdown on some PowerBook 170's one gets a spurious BusError in *The Debugger*.
- Attempting to set a breakpoint at an address > 16Meg that is not in a task will not work.
- Using MMU protection with programs built with Model FAR causes spurious errors, see DTN #7.
- Using the Think C Debugger in combination with *The Debugger* causes programs to stop at the end of The Debugger's patch to _LoadSeg as the TC Debugger sets `LoadTrap` (byte at $12D).
- ***The Debugger* does not work with System 7 VM**
- Do not try to use a Syquest with the R45 INIT as a boot disk, *The Debugger* will screw up.
- The Debugger does not handle .SYM file info for arrays with variable bounds.emitted by LS Fortran.

ROMinations

Over the years Apple has produced a number of ROMs. Aspects of the ROM which are important to The Debugger are the base address of the ROM, which is in the low memory global ROMBase, and the version which is in the word at ROMBase+8. Apple has produced the following major versions of the ROM:

Size	ROM Id	.snt	Used On
64K	??	NO	128K & 512K Mac's - **NOT** supported by The Debugger
128K	0075	BlueDisk	Mac +, Mac Classic is really 128K ROM with resources
256K	0178	BlueDisk	Mac II, IIx, IIcx, SE/30
256K	0276	BlueDisk	Mac SE
256K	037A	build	Old Portable (Luggable)
512K - 1Meg	067C	Default*	Mac IIci, IIfx, PowerBooks, Quadras, etc
2Meg	077D	build	Centris 660AV & Quadra 840AV (Tempest & Cyclone)

The 067C ROM is one of the most popular ROMs. It is the first 32 bit clean ROM.
It implements System 6.0.4 in it. Previous Mac II ROMs are 32 bit dirty, and are poor choices for debugging, or use with an add-on accelerator board.
The standard base address for the 067C ROM is $4080 0000. Some oddball Mac's like the IIsi and the LC use a base address of $40A0 0000, which mean that the supplied ROM .snt doesn't work.

The 077D ROMs in the new Macs implement System 7.1 in the ROM, reducing the size of the patches required to run. The 077D ROMs will be the basis for most future Macs.

68040 Notes

The Jump trace bit does not work correctly in the 68040 CPU chips (Motorola screwed up). As a consequence, the 'Trace Jumps' and 'Proc Entry/Exit Trace' commands do NOT work on Macs with these CPUs.

Future (Work in Progress) - PowerPC

While many of you will have a fairly easy conversion over to the PowerPC Macs, I have quite a bit of work to do in order to get The Debugger to debug PowerPC programs in 'Native' mode.
As part of the effort, I have to understand the hardware, the software, write new code for many of The Debugger features such as walking the stack, etc. I hope to have something available late this fall (by December ??).
Given this fact, I suspect that MMU protection for 040's will never happen.

Sincerely,

Steve Jasik **AppleLink: D1037 Compuserve ID: 76004,2067**

Debugger Tech Note #11

Parser Extensions & 'dcmds'

August 93

The basic expression language is described on pages 37 to 40 of the gray manual and summarized in the '-D_Help-' file which is displayed when the Help command is selected in The Debugger.
This tech note discusses additions to the expression language that are not in the gray manual.
They make it useful for the writing of scripts which display the contents of tables, lists, etc.

Output Re-direction

Normally commands, write statements in 'action clauses', etc put their output at the end of the '-Notes-' window. The ?ReDirect command lets one define an alternate window for such output.
The syntax is:
?ReDirect('windowName' {,Append}); - If 'windowName' exists then bring it to the front, else create a window with a name 'windowName' . If the Append parameter is present, then output will be appended to the end of the text in the window, else the text will be deleted and the window drawn as empty the next time it is written to.
?ReDirect; - re-directs output to the '-Notes-' window in Append mode.

Note: Unlike MPW, The Debugger always redirects output to the end of text in the redirection window.

Write Statement Extensions

The syntax of a list in a write statement is:

```
write_list := List_ele { ,List_ele }  0 or more times.
List_ele := 'string_constant' | ?REC(TypeName,address) | expression { :TypeName {:width} }
```

The extension is that we can now control the width of the output field by specifying it.
For example, write(?xx:HLongint:6); displays the value of the Debugger variable ?xx in a field of 6 columns instead of the normal 10 columns (blank, dollar-sign and 8 digits). It does this by truncating the leading 4 columns from the output.
This extension enables us to display tables in neat columnar format.

Array References

You may reference the elements of a 1-dimensional array via the notation: ArName[subScript_expr] .
If ArName is a pointer to an Array, then one writes: ArName^[subScript_expr] and so on.
The address is calculated as:

AddrOf(ArName) + sizeOf_Arname_element * (subScript_expr - Lower_bound)

which is just what you would expect. For an example of it's use see the 'Files' script in 'Dbgr Scripts'.
You can reference any of your global variables in your program during the execution of a script.
The way to display an element of an array of Extended elements is: write(myArray[n]:Extended);

While, Repeat-Until loops and CYCLE and LEAVE statements

WHILE and REPEAT - UNTIL loops are implemented just as they are in Pascal.
You may nest loops up to 8 levels deep.
A CYCLE statement will cause a branch to the end of the current loop and a LEAVE statement will branch to the statement following the end of the loop.

'Dbgr Scripts' contains a number of examples of WHILE loops. FOR loops are not implemented.

The ?UserC command

The ?UserC command was created for those of you who want to dump more intricate structures then one can easily do with the expression language. An example is the symbol table of a compiler.

The syntax of the command is: `?UserC(expression, ProcName);`
Expression will be evaluated and the 32-bit value passed to the procedure ProcName in your program. The output of `printf` statements from ProcName will be intercepted by The Debugger, buffered up and displayed in a window when the ?UserC command completes.

`printf` and all the procedures it calls must be loaded when the command executes, else _LoadSeg will be called and it will cause the value of the PC and some of the registers to change, which is NOT what you want.

A Sample script that calls of ?UserC is:

```
?Redirect('SymTab Dump');   ?UserC(SymTab_Ptr,dump_symbol_tbl);  ?ReDirect;
```

As always, the above line can be part of an action clause, or one can triple-click on the above line and press 'Shift-Cmd-[' (Shift-Calculate).

Macsbug 'dcmds'

After extending the expression language and writing some sample scripts with it, I feel that there are very few 'dcmds' that are worth using. Two are 'Leaks', the Memory leak detector by Bob Johnson, which is available on the Apple Tool Chest Developer CD-ROM and the 'thing' dmcd which displays the loaded QuickTime components.

Dcmds may be loaded at Debugger initialization time by adding the following lines to ROM.dsi:

```
=Z  ; Load Resident dcmd's
file_name of a resource file containing 'dcmd' resources
```

(you may specify multiple file names on separate lines).
If you don't specify a full pathname, then the dcmd files should be in the 'Nosy/Dbgr files' folder.

`?dcmd(dcmd_name,'argument');` will execute a 'resident' dcmd.
`?dcmd;` will list all the resident dcmd's and their 'help' prompts.

Dcmds may also be loaded at **any time** and executed via the statement:

```
?dcmd(dcmd_name,'argument',pathname);
```

I have only implemented a subset of the dcmd callbacks described in the Macsbug Reference and Debugging Guide so you are limited to passing only one argument to a dcmd. While the dcmd will be given an initialization call, it may not work properly in the case that you load the dcmd at other than Debugger initialization time. The Leaks dcmd must be loaded at Debugger Init time.

ACI US ships Object Master™ Universal version 2.0, a source code editor and development tool.

ACI US, Inc. is currently shipping Object Master™ Universal version 2.0, an upgrade to its dynamic source code editor and development tool. The new version provides significant enhancements to facilitate the development of applications created in the MPW®, THINK C™, Symantec™ C++, and THINK Pascal™ environments.

Object Master's Project window allows developers to specify which resources to include in a project. As files are added, Object Master parses them, creating a dictionary of component file information. The dictionary includes information such as definition and location of all classes, methods, procedures, fields, and data types.

Using the Browser window, developers can view the entire project through lists of procedures, functions, classes, methods, and fields. Developers can also access specific sections of code by selecting any method or procedure in the Browser window. Editing can be done in either the Browser window or the File window. When a function or method implementation changes, Object Master updates the header automatically, eliminating the need for double-editing. When the developer adds a new method to a class, Object Master automatically creates a skeletal implementation for it.

Object Master's full-featured source code editor was written by programmers for programmers. The editor provides programming-specific features such as:

Syntax checking: Object Master checks a file's syntax when it is added to the project or when the developer saves changes. Automated syntax checking reduces the need for frequent compiling.

Function call template generation: Developers can use template calls generated automatically by Object Master to call any method or function.

Custom formatting: When displaying a file from a supported language, Object Master color codes and stylizes text for easy identification. Developers can customize styles for each of the six language elements: procedures, keywords, compiler directives, syntax errors, disabled directives, and comments.

Automarkers: Markers created by Object Master for every procedure and function in a project can be used to quickly locate source code. Errors are indicated by markers which help users identify and access problematic code.

The Segment window lists all segments in the project and displays the methods and procedures each segment contains. In this window, developers can easily resegment source code by dragging and dropping methods and procedures into segments.

In the Class Tree window, Object Master graphically depicts a project's class and provides easy access to methods, procedures, and fields within the hierarchy. This window facilitates navigation by depicting the hierarchy of a project and displaying all methods and fields. Clicking methods and fields in this window allows users to display the selected routine for editing.

With version 2.0, Object Master introduces over 150 new features and enhancements. Object Master Universal version 2.0 offers improved speed of operations. For example, parsing speed has increased two to three times in the new release. Another parser enhancement is its extended code support—the new version can parse C++ 3.0 code and, thanks to its support for macros, can even parse non-Macintosh code. For quick and uninterrupted parsing, parser errors are placed in an error list that is displayed when parsing is completed.

With version 2.0, Object Master Universal offers users extended growth potential and expansion paths. The new version works seamlessly with MPW, THINK C, Symantec™ C++, and THINK Pascal, allowing users to create applications using multiple development environments. No additional modules are necessary—communication between Object Master and the development environments is performed via AppleEvents. Additionally, the new version continues to support procedural programming as well as object-oriented programming, offering users a smooth transition path from purely procedural programming to object-oriented programming.

Other new features in Object Master Universal version 2.0 include modeless search dialogs, project history lists, filters for class/file displays, and support for Apple's SourceServer and AppleScript™. The AppleScript support allows users to submit queries on the project, auto[illegible] common tasks, and add custom menus. Object Master Universal surpa[illegible]ere source code editors by offering additional features that support users when they need it most—during code development.

To Order Object Master™ Universal, call us at (408) 252-4444 x252

For more information or a demo diskette, call (408)252-4444

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

March 17, 1993

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of the Universal Version of *The Debugger V2/93* and *MacNosy* and some accompanying notes on them. This release contains numerous bug fixes and new features.

For those of you who purchased *The Debugger* prior to December 1, 1992, this is you last free update. To receive updates and support in 1993, fill out the enclosed invoice and return it to me before May 1, 1993 with authorization to bill your credit card or a check for $130 (payable to a US Bank).

Tech Support for The Debugger is available via phone, Applelink (D1037), or CompuServe (70277,776).
Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) Plus sales tax and $10 shipping.

General
Site Licensees - The floppy contains a stuffed copy of this cover letter in MS Word 4 format.
Is your address label correct?? Is The Debugger/MacNosy meeting your needs, if not complain to me.

68040 Notes
The Jump trace bit does not work correctly in the 68040 CPU chips (Motorola screwed up). As a consequence, the 'Trace Jumps' and 'Proc Entry/Exit Trace' commands do NOT work on Macs with these CPUs.

INIT notes
You may bypass most problems with INITs by loading The Debugger early. To do this, rename 'xDbgr_Startup' to '!xDbgr_Startup' or select it to load before any INITs, particularly NOW Utilities V4. This will also get around problems with WorldScript, CD-ROM V4, etc, etc.
xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
DPI on the Fly - move to it to Extensions and make an Alias to it for use as a Control Panel.
DiskLocker's (DiskLock, etc) appear to foul up the operation of *The Debugger*.
MicroSoft Mail must load **after** *The Debugger*.
Do NOT use any form for 'Mr Bus Error', 'evenBetterBus error', etc., as it conflicts with *The Debugger*.
SuperBoomerang- use V3.01 or later as some of the 3.0 versions had bugs.
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger*.
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
INIT_ADB is an INIT to activate the 'interrupt' (Command-power) key on the IIsi and Mac LC machines. It is in the folder: '`Misc INITs & Source`'.

Installation Procedure
This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
The ROM/CODE.snt for the Mac IIci/IIfx,... **was omitted**, get a copy from an older release.
If you have MPW, then I strongly suggest that you use the 'Install Debugger' script .
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
Copy or create a ROM .snt for your machine (default in Dbgr/Nosy files is for a 256K ROM Mac II)
Boot the mac, and validate that the new version is operational.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

System Compatibility
The Debugger and *MacNosy* have been tested on Systems 6.0.5, 6.0.7 7.0 , 7.0.1, 7.1, & the System 7 TuneUp. For the Classic and LC you must build a ROM/CODE.snt file from scratch. I have also tested *The Debugger* on the Centris 610,Centris 650, Quadra 800 and PB 165c, and some unannounced machines.

MPW's *ILink* vs *The Debugger*
I will NOT be supporting the State (.NJ) files produced by ILink which is part of ETO #10.
I feel that IBS is still a superior solution, and urge you to give it a try if you haven't already.
This release of PatchLink contains bug fixes to make it work with big segments, big jump tables, and fixes the problem of lost '^^name' types.
For your info, I have included some recent mail from AppleLink on the subject on the disk.

Jasik Designs

343 Trenton Way Menlo Park, CA 94025 (415) 322-1386

July 27, 1992

Dear Mac II Nosy/Debugger User,

Enclosed is the latest release of the Universal Version of *MacNosy* and *The Debugger V2/92*, and some accompanying notes on them. This release contains numerous bug fixes and new features.

Tech Support for The Debugger is available via phone, Applelink (D1037) or CompuServe.

Site (multiple copy) Licenses are available through Jasik Designs. The pricing is $225 per copy times the number of copies (programmers using the product) plus sales tax and $20 shipping.

68040 Notes

The Jump trace bit does not work correctly in the 68040 CPU chips (Motorola screwed up), and as a consequence, the 'Trace Jumps' and 'Proc Entry/Exit Trace' commands do NOT work machines with these CPU's. **Motorola DOES NOT intend to fix the problem. Because of this and the fact that MMU protection works better on the 68030 Macs (IIci, IIfx), they are still the best machines for debugging.**

INIT notes

MicroSoft Mail must load **after** *The Debugger*.
SuperBoomerang- use V3.01 or later as some of the 3.0 versions had bugs.
After Dark users should be using version 2.0V or later, it should load **after** *The Debugger*.
xDbgr_Startup **MUST** be placed in the Extensions folder on System 7.
NOW utilities Startup Mgr - **disable the options**: 'protect screen from erasure' & 'display invisible INITs'
INIT_ADB is an INIT to activate the 'interrupt' (Command-power) key on the IIsi and Mac LC machines. It is in the folder: '`Misc INITs & Source`'.

Installation Procedure

This update is shipped as a Self Extracting Archive (.sea) file. Double click on it to unpack the files.
The ROM/CODE.snt for the Mac II /IIx/IIcx,... **was omitted**, get a copy from an older release.
If you have MPW, then I strongly suggest that you use the 'Install Debugger' script.
To manually install *The Debugger*, do the following:
Copy the files in the "put files in System fldr" to your System folder. **You must do this step.**
Copy or create a ROM .snt for your machine (default in Dbgr/Nosy files is for a 512K ROM Mac IIci/IIfx...)
Boot the mac, and validate that the new version is operational.
There are no significant changes to the MacApp of IBS mods in this release and you may omit changing them.
If you are using Version 3 of MacApp, then merge the IBS mods into the IBS folder.
If you are using IBS, then reinstall it, using the IBS_Install script.

System Compatibility

The Debugger and *MacNosy* have been tested on Systems 6.0.5, 6.0.7 7.0 , 7.0.1, 7.1, & the System 7 TuneUp.
For the Classic and LC you must build a ROM/CODE.snt file from scratch.

Miscellaneous Notes

- You may now **change command key equivalents** in The Debugger via the '`-Menu`' command in the .dsi file, see the documentation in ROM.dsi for details. Now you can close windows with Cmd-W !!
- Cmd-D - if a partial name (Class::mm) is typed in, then open up the procedure that contains that name OR list all proc names that 'match' in a window.
  For example, if `Txyz` is a class, then entering "`txyz::`" into the Cmd-D box will list all its methods.

I have been successful in shipping patches to The Debugger to some you this spring via AppleLink, and will continue to do so this summer. Shortly after the disks were duplicated, I fixed a significant bug in SYM file type processing. The fix will be sent to users with AppleLink ID's only. I can NOT send binary files across to Internet addresses. Send me your AppleLink ID if you haven't already done so.
Patch files sent out this summer will use this version (7/23/92) as the base version.

Debugger Tech Notes (DTN's) are on the floppy in MicroSoft Word 4 format along with a table of contents. DTN's 0, 5, 6, 7 and 9 have been revised and DTN 10 on Performance timing is new.

New Features and Fixes, etc (see the "Change History" file for a complete list)

- CoverTest V.99.5 is included. A short user guide for it is in the 'Dbgr Tech Notes' folder.
- Add a '****' FREF so Nosy is System 7 'Drag and Drop' compatible
- Add a Performance Timing command & windows (see DTN #10 for details)
- Shift-'Heap Scramble' scrambles the System Heap (no auto turnoff)
- add ability read protect 0 page of memory in MMU Protection
- add ability to use MMU Protection for MPW Tools
- Dbgr is configurable for Foreign keyboards etc in ResEdit using the 'Dcfg' resource
- Heap display - move summary to front & display Detached Resources (file refnum = '<??>')
- add Dbgr flag 'Sgned_Char_Strs' to force 'Array of char' to be a string
- Increase size of trap patch table from 512 to 768

fix - MMU Prot on & Heap Scramble => spurrious BusErr's from temp buffer used by HS
fix - Dbgr hung in I/O System when Watchpoint tripped during a SCSI Read
fix - to get along with V1.3 of RadiusWare for their 040 accelerators
fix - if Univ ROMs then skip NotifyMgr call in _SystemTask in Dbgr so we can debug them
fix - to work with QuickDraw GX
fix - undo 1/9/92 fix to processing of Local Vars that caused incorrect display of Pascal VAR params.
Also add logic to delete dup displays of mem & reg values if both are the same

Known Problems

- The text drawing routines put garbage on the user's screen when using Step Continuous.
- When doing a Shutdown on some PowerBook 170's one gets a spurrious BusError in The Debugger.
- Attempting to set a breakpoint at an address > 16Meg that is not in a task will not work.
- Using MMU protection with programs built with Model FAR causes spurrious errors, move the CODE seg named '32-bit bootstrap' to the System heap to avoid them. Use the following Shell command:

```
echo "change 'CODE'(∂"32-bit bootstrap∂") to $$type ($$id,$$Attributes|64 );" ∂
  | rez -a -o myAPPL
```

- MPW C outputs bad SYM info for enum declarations that contain refs to symbols previously defined in the enum (enum {a,,b, c = a | b};). The Debugger may blowup during type processing. A fixed C compiler should be available by ETO #8 time. QuickTime users should fix Movies.h.
- Using the Think C Debugger in combination with The Debugger causes programs to stop at the end of The Debugger's patch to _LoadSeg as the TC DEbugger sets `LoadTrap` (byte at $12D).
- The MDEF distributed with IBS for use in the MPW Shell **does not** work with System 7.
- **The Debugger does not work with System 7 VM**
- Do not try and use a Syquest with the R45 INIT as a boot disk, The Debugger will screw up.
- MMU protection does NOT work on Mac II's with the Dove Accelerator installed.
- The Debugger does not handle .SYM file info for arrays with variable bounds.emitted by LS Fortran.
- some very large programs have more than 65K global vars which exceeds the limit of the SYM file format. There are NO warning messages from the Linker or Debugger but garbage names in the Local Vars display.

Future (Work in Progress)

We will be working on the following features for the next release (October 1992):

- MMU Protection for Macs with 040 CPUs

Shows I will Attend

I will be at the Boston Macworld in August (at the Apple Tools expo).

The Head Nose on CD-ROM/QuickTime

Is in progress and will be done by this fall hopefully.

Sincerely,

Steve Jasik **AppleLink: D1037** **Compuserve ID: 76004,2067**

Debugger Tech Note #10

Performance Timing Analysis

July 92

Control

Information

The Debugger now contains a timing facility that is accessible via the "Performance Timing ..." command in the Go menu. The facility monitors program execution and presents a histogram of where the program is spending its time. The command will work on all Macs from the Mac II on; it has the following selections:

1) Timing is OFF:
You will be presented with a type-in dialog box and the '-Tasks & Files-' window.
Type in the number of the task you want to monitor and press return. The task must be either an Application or MPW Tool. When you exit The Debugger, it starts a timer task that samples the program counter (PC) at regular intervals and builds a histogram of it in memory.
When you next enter The Debugger, the histogram results will be displayed in two windows, titled:
`'Perf Time - Raw Stats'` and `'Perf Time - Sorted info'`.

2) Timing is ON:
You may turn timing OFF by selecting the "Performance Timing ..." command, and will get a message in the '-Notes-' window that it has been turned off,
Or you may clear the buckets by holding down the shift key and selecting the "Performance Timing ..." command. In this case, the next entry to The Debugger will generate windows with the names:
`'Perf Time - Raw Stats - 1'` and `'Perf Time - Sorted info - 1'`.

Some Details

When the performance timer is active, the PC is sampled at regular intervals (500 microseconds on Macs with 040 CPUs and 1 millisecond on all others), and the value of the appropiate timing 'bucket' is incremented. Each bucket covers a 16 byte range. Attempts to increase the sample rate by decreasing the interval would increase the time to run the performance analysis as the overhead due to the time manager is excessive. Twenty years ago we were able to sample the PC at a rate of 30K samples per second on a 3 MIPS CPU without disturbing the program.
Today it is difficult to get 2K samples per second on a 15 MIPS CPU. Such is progress !!
The amount of memory required for the buckets is :
(system_heap_size + ROM_size + Zone_size_of_monitored_task) divided by 8 (≈ 400K or more)
The space for the buckets is allocated in MultiFinder's heap zone.

When the Timing analyzer is active, unloading of segments is disabled so that the 'Raw Stats' window contains valid addresses. Your program may need more than the normal amount of space to execute.

Interpreting the Results

The 'Sorted Info' window lists of the 60 most frequently executed procedures in descending order. An example of it is to the right:

```
---Hits---- % ----name--
 15621   29.62%  ∂5.FIND_PROC
  3086    5.85%  ∂5.PUSH
  2630    4.98%  ∂5.FIND_BLK_LIMIT
  2153    4.08%  ∂5.MOVEB_UP
  2139    4.05%  ∂5.PUSH_LJ
  1716    3.25%  ∂5.DECODE_INST
  1414    2.68%  ∂5.FIND_HOLES
  1277    2.42%  ∂5.SRCH_LAB
```

The columns are Hits (PC samples in the procedure) the percentage relative to the total number of samples, and the name of the procedure preceded by the task number (∂n).

The '**Raw Stats**' window is a bit more complicated, it contains a header and three sections. One for the task, one for ROM and one for the RAM patch area (in the System heap). An example of the window is:

```
hz@4FCC30  NosyII

RAM  hits    748     1.41%            summary lines
ROM  hits  10158    19.26%
Task hits  41821    79.31%

•Task bucket size = 16
 Address--Hits-Offset-Rsrc-Id/Proc
=04FD190   130     2E ∂5.LIST_HEX
=04FD1A0   375      4 ∂5.ADDCHAR
=04FD1B0    84     -4 ∂5.OUT_DIG
=04FD250     1     -4 ∂5.UPPEFCASE
=04FD270    52     1C
=04FD3A0    41     38 ∂5.SORT
=04FD3B0    52     48
  . . . .
=0500780     9     -6 ∂5.FIND_PROC     (current proc name, bucket start = proc-6)
=0500790     7      A
=05007A0  9494     1A
=05007B0  6107     2A    ( 6107 samples in the bucket starting at FIND_PROC+$2A )
=05007C0     4     3A
=0501820     1      2 ∂5.FIND_CASE_INFO
  . . . .
•ROM  bucket size = 16
 Address--Hits-Offset-Rsrc-Id/Proc
40800000   507     -4 ∂0.rDA_5             (1st ROM bucket records samples
40804BE0     9      8 ∂0.X_SwapMMUMode      above the task & below ROM )
40804C10   126     38
4080A2A0    20      A ∂0.Lvl1_int1
4080A2B0     4     1A
```

The columns in the window are the address of the bucket, the number of samples in the bucket (16 byte address range), the offset from the start of the current procedure, and the name of a procedure or resource.

To go from the 'sorted info' window to the proc name in the Raw Stats' window, hilite a '`∂n.procName'`, copy it, switch windows, and Cmd-G to find the info for the procedure in the 'Raw Stats' window.

To view the source/object code, hilite an address and Cmd-D. If you have a source window, use Cmd-Click to switch to the interspersed assembly list view.

Using the Timing Facility

Enter The Debugger and set breakpoints at points in your program where you want to turn the timer on and off. When you reach the start point, select the "Performance Timing" command to turn the timer on.
When you reach the stop point, you can view and analyze the results, create text files for later use, etc. Remember that the addresses may not be valid at a later time if you have unloaded segments, etc.

If you are timing MPW Tools, then build a MPW Shell .snt file in Nosy so you see the time spent in the Shell.

Use the Performance timing facility to find critical procedures in the program. If you insist on fine tuning an inferior algorithm (like a bubble sort), then the effect of using the timing facility may be minimal.
On the other hand, if you use the results to think about the algorithm, then you may end up scrapping it, and replacing it with one that is more effective.

In the sample windows above, I did a timing analysis on the treewalk in Nosy and found that most of the time was spent in a linear search routine. As a quick band-aid, I recoded it in assembly language (all of 10 lines), and cut the percentage of time spent in that routine by a factor of 2. A better solution would be to keep the table that is being built sorted so that existing binary search routines could be used.

You should attempt to do timing analysis on the parts of your program that do not do heavy disk activity as you may find that a large amount of your time is spent in loops waiting for the SCSI requests to complete.
An alternative is to move the files you will be accessing to RAM Disk.

```
Item forwarded by       D1037        to JOHAN

Item forwarded by       D1037        to CDA1051

Item forwarded by       D1037        to INTRIGUE.DVJ
                                        KILROY

Item forwarded by       D1037        to D5988
                                        IRIS

Item forwarded by       D1037        to ORACLE.NTD
                                        G.CONNELL
                                        SCREENPLAY
                                        KODAKEN
                                        WP.WORKS
                                        V0419

Item forwarded by       D1037        to D3423
                                        BEZANSON
                                        ZULCH.R
                                        LIFETOUCH

Item forwarded by       D1037        to CDA0425

Item forwarded by       D1037        to ABACUS
                                        D2275

Item forwarded by       INTELLIGENCE to IAL

Item forwarded by       D1037        to AUST0427

Item forwarded by       D1037        to COREL
                                        ANDREW.ENG
                                        CDA0195
                                        V0645
                                        BRINKMANN
                                        STAT.EASE
                                        ADOBE.BHAST

Item forwarded by       D1037        to FORETUNE.DVJ
                                        MADA.LIB

Item forwarded by       D1037        to HOL0146

Item forwarded by       D1037        to SW0159
                                        FALK

Item forwarded by       D1037        to D2453

Item forwarded by       D1037        to GER.XUU0003
```

Sub: 6/23 Dbgr patch files

Enclosure: 6-23DbgrPtch.PKG

6/23 Jasik Debugger patch Notes - June 23

To All,

Enclosed is a patch file to The Debugger and a newer version of
XDbgr_Startup.

A version of CoverTest that works with Think C PROJects is available
seperately (Link me and request it).
You will need this version if you are using CoverTest.

I will be mailing out an update disk around mid July.

••• The 6/21 patch file had a bug in it relative to the handling of
ASLM libraries, hence this patch file.

In addition, I have added the following:

- add names of static funcs to ThC PROJ if 'ThC_Static' flag is ON
  AND source is available

- The value $fffDfffD for Bad Zero does NOT work well for 040 Mac's
  so The Debugger now uses $50ff50ff for those Mac's
  This change does NOT apply to Accelerated Mac's

IN addition to the changes since 5/24 I have done the following to
The Debugger:

- revise handling of ASLM so that one can debug Multi-seg Libraries.
  Also, if resource type to be debugged is NOT = 'code', then SYM file name
  must be of the form 'LibraryName.rrrr.SYM' where rrrr is the Resource type.
  Tasks are deleted when all segments are unloaded.

fix - System hangs when Dbgr, FileShare and ASLM active

fix ? - make _SystemTask, etc a NO-OP inside The Debugger to
promote better operation with AOCE, WorldScript, etc

fix - add a KLUDGE to display the 1st 9 chars of a '^SignedByte' for C programm
This is necessary as C doesn't have a well defined String type.

fix - blowup in Tbl mgr when trying to increase size of NTE_ITT (GT_Add_Used)
fix - Paste in Find/Change dialog didn't work
fix - misc step into bug in Dbgr caused garbage display of Trap call window
fix - limit of one beep when errors in Heap display

Fixs in the 5/24 package were:

work with WorldScript II independent of load order.
Work wih AOCE

fix - spurrious heap errors when both Heap Check & MMU protection selected
fix - ThC projects - failure to test for end of procs caused memory to be munge
fix - allow writes to GhostWindow with MMU protection ON
fix - Local Vars display - validate current A6 before trying to display Local V
fix - more than 16K types, then didn't re-read SYM.snt

To update your version of The Debugger, change the name of
of the 3/18/93 version of 'The Debugger' to:

'3/18 Debugger' and dbl-click on the UpdateMaker
file '6/23 - Dbgr Patch' to create a new version
of 'The Debugger' (in the 'Dbgr/Nosy files' folder).

Steve Jasik

----Users of Think C Reference V2:

A patch file to create Version 2.0.1 which is compatable
with The Debugger is available in the :
'Third party Updates:Symantec Solutions:Updates...'
area of AppleLink.

OBJECT MASTER

What is Object Master?

Object Master is a tool designed for programmers using MPW (C, C++, and Pascal), THINK Object Pascal and C, Modula 2 - P1, and other programming languages on the Macintosh. Object Master greatly increases programmer productivity by providing intelligent source code editing, object browsing, resource browsing, segmentation mapping, and file mapping all in an environment fully integrated with the above programming languages via Apple Events.

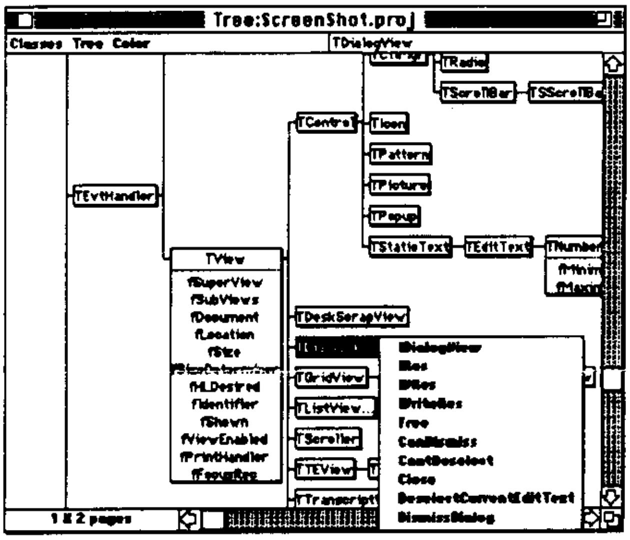
Intelligent source code editing

One example of intelligent source code editing is the ability of the Object Master editor to check syntax on-the-fly, catching most errors without having to compile. As you create your files, a marker list is automatically generated and provides easy access to all procedures and objects that have been defined in the file.

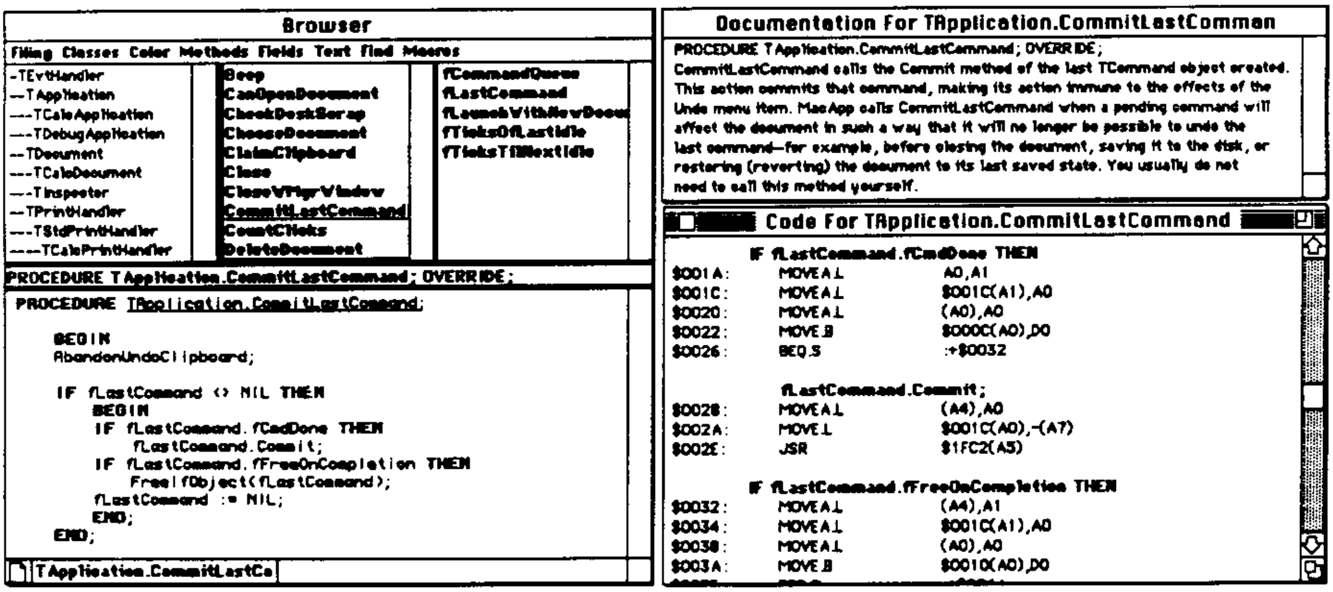
With Object Master, you can define custom code templates that are used to check the consistency of your personal coding style. The Object Master editor also allows you to assign unique colors and styles to procedures, keywords, and templates. Automatic indenting is also supported by Object Master.

Organize your files into a project

Object Master automatically maintains a database of data definitions and procedures. This organization of your files provides for fast navigation and structure editing. An Object Master project can mix Pascal, C, Modula, Rez, and 411 files. For example, a Pascal program that calls several C libraries can be edited as a single project with Object Master.



Class Tree Window provides graphical display and navigation through the project's class hierarchy.



```
-TEvtHandler
--TApplication
---TCalcApplication
---TDebugApplication
--TDocument
---TCalcDocument
---TInspector
--TPrintHandler
---TStdPrintHandler
----TCalcPrintHandler
```

```
Beep
CanOpenDocument
CheckDeskScrap
ChooseDocument
ClaimClipboard
Close
CloseWMgrWindow
CommitLastCommand
CountClicks
DeleteDocument
```

```
fCommandQueue
fLastCommand
fLaunchWithNewDocu
fTicksOfLastIdle
fTicksTillNextIdle
```

```
PROCEDURE TApplication.CommitLastCommand; OVERRIDE;
```

```
PROCEDURE TApplication.CommitLastCommand;

    BEGIN
    AbandonUndoClipboard;

    IF fLastCommand <> NIL THEN
        BEGIN
        IF fLastCommand.fCmdDone THEN
            fLastCommand.Commit;
        IF fLastCommand.fFreeOnCompletion THEN
            FreeIfObject(fLastCommand);
        fLastCommand := NIL;
        END;
    END;
```

TApplication.CommitLastCo

Documentation For TApplication.CommitLastCommen

PROCEDURE TApplication.CommitLastCommand; OVERRIDE;
CommitLastCommand calls the Commit method of the last TCommand object created. This action commits that command, making its action immune to the effects of the Undo menu item. MacApp calls CommitLastCommand when a pending command will affect the document in such a way that it will no longer be possible to undo the last command—for example, before closing the document, saving it to the disk, or restoring (reverting) the document to its last saved state. You usually do not need to call this method yourself.

Code For TApplication.CommitLastCommand

```
            IF fLastCommand.fCmdDone THEN
$001A:      MOVEA.L         A0,A1
$001C:      MOVEA.L         $001C(A1),A0
$0020:      MOVEA.L         (A0),A0
$0022:      MOVE.B          $000C(A0),D0
$0026:      BEQ.S           :+$0032

               fLastCommand.Commit;
$0028:      MOVEA.L         (A4),A0
$002A:      MOVE.L          $001C(A0),-(A7)
$002E:      JSR             $1FC2(A5)

            IF fLastCommand.fFreeOnCompletion THEN
$0032:      MOVEA.L         (A4),A1
$0034:      MOVEA.L         $001C(A1),A0
$0038:      MOVEA.L         (A0),A0
$003A:      MOVE.B          $0010(A0),D0
```

A Documentation Window and a Code Display Window can be linked to a Browser Window to provide automatic access to any 411 documentation for the method being edited as well as a disassembled view of the method.

OBJECT MASTER

What is Object Master?

Object Master is a tool designed for programmers using MPW (C, C++, and Pascal), THINK Object Pascal and C, Modula 2 - P1, and other programming languages on the Macintosh. Object Master greatly increases programmer productivity by providing intelligent source code editing, object browsing, resource browsing, segmentation mapping, and file mapping all in an environment fully integrated with the above programming languages via Apple Events.

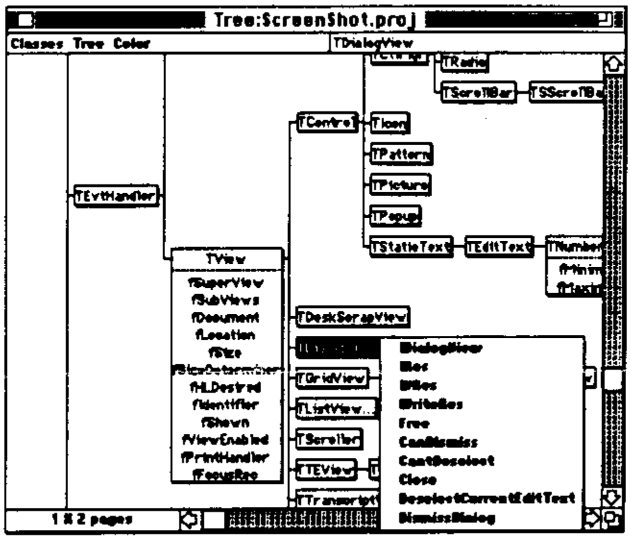
Intelligent source code editing

One example of intelligent source code editing is the ability of the Object Master editor to check syntax on-the-fly, catching most errors without having to compile. As you create your files, a marker list is automatically generated and provides easy access to all procedures and objects that have been defined in the file.

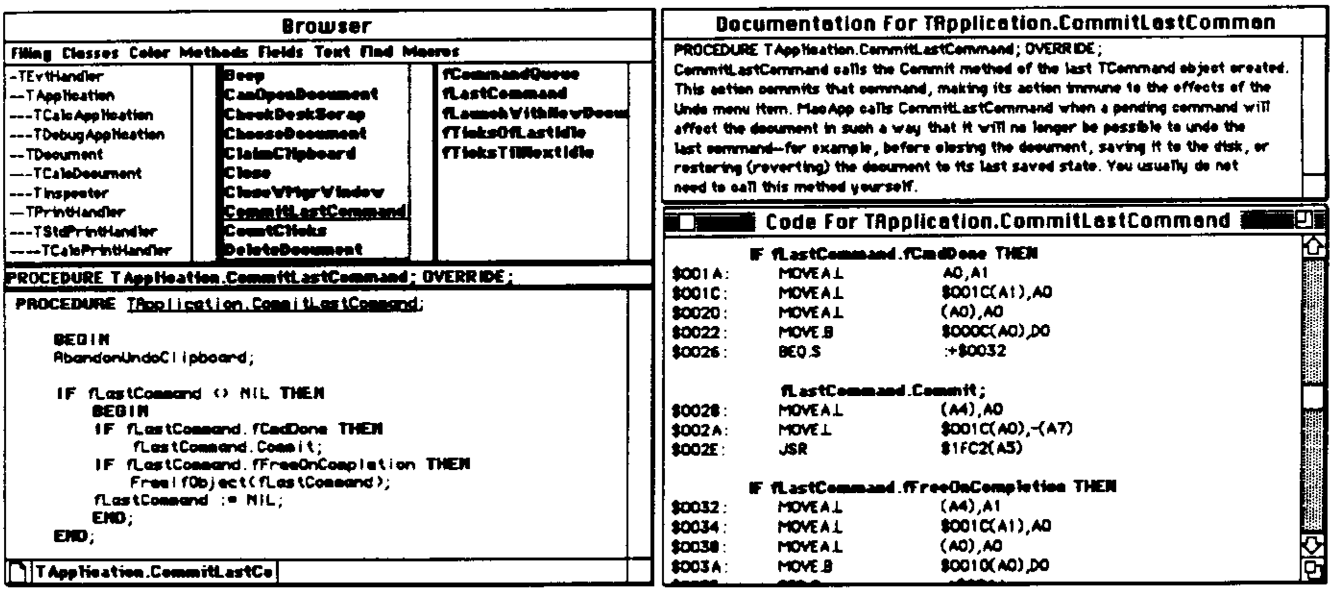
With Object Master, you can define custom code templates that are used to check the consistency of your personal coding style. The Object Master editor also allows you to assign unique colors and styles to procedures, keywords, and templates. Automatic indenting is also supported by Object Master.

Organize your files into a project

Object Master automatically maintains a database of data definitions and procedures. This organization of your files provides for fast navigation and structure editing. An Object Master project can mix Pascal, C, Modula, Rez, and 411 files. For example, a Pascal program that calls several C libraries can be edited as a single project with Object Master.



Class Tree Window provides graphical display and navigation through the project's class hierarchy.



```
PROCEDURE TApplication.CommitLastCommand;

    BEGIN
    AbandonUndoClipboard;

    IF fLastCommand <> NIL THEN
        BEGIN
        IF fLastCommand.fCmdDone THEN
            fLastCommand.Commit;
        IF fLastCommand.fFreeOnCompletion THEN
            FreeIfObject(fLastCommand);
        fLastCommand := NIL;
        END;
    END;
```

Documentation For TApplication.CommitLastComman

PROCEDURE TApplication.CommitLastCommand; OVERRIDE;

CommitLastCommand calls the Commit method of the last TCommand object created. This action commits that command, making its action immune to the effects of the Undo menu item. MacApp calls CommitLastCommand when a pending command will affect the document in such a way that it will no longer be possible to undo the last command--for example, before closing the document, saving it to the disk, or restoring (reverting) the document to its last saved state. You usually do not need to call this method yourself.

Code For TApplication.CommitLastCommand

```
          IF fLastCommand.fCmdDone THEN
$001A:      MOVEA.L        A0,A1
$001C:      MOVEA.L        $001C(A1),A0
$0020:      MOVEA.L        (A0),A0
$0022:      MOVE.B         $000C(A0),D0
$0026:      BEQ.S          :+$0032

            fLastCommand.Commit;
$0028:      MOVEA.L        (A4),A0
$002A:      MOVE.L         $001C(A0),-(A7)
$002E:      JSR            $1FC2(A5)

          IF fLastCommand.fFreeOnCompletion THEN
$0032:      MOVEA.L        (A4),A1
$0034:      MOVEA.L        $001C(A1),A0
$0038:      MOVEA.L        (A0),A0
$003A:      MOVE.B         $0010(A0),D0
```

A Documentation Window and a Code Display Window can be linked to a Browser Window to provide automatic access to any 411 documentation for the method being edited as well as a disassembled view of the method.

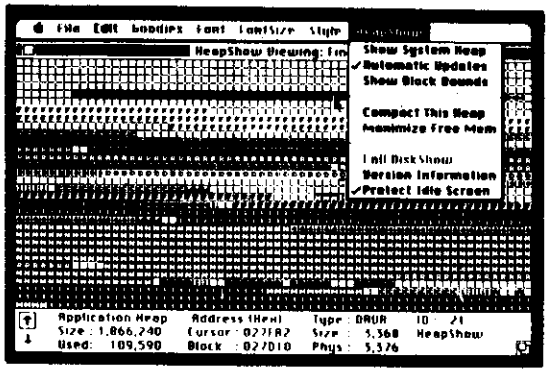
HeapShow™

may chan[:he old proverb to "a picture is worth a thousand pages," by showing graphically what it takes a popular Macintosh™ book two entire volumes to reveal.

Graphically represents different memory blocks.

Mouse or keyboard controlled for more specific information.

Scrollable window for viewing any size heap.



Application Heap
Size : 1,866,240
Used: 109,590

Address (Hex)
Cursor : 027FA2
Block : 027D10

Type : DRVR
Size : 5,360
Phys : 5,376

ID : 21
HeapShow

HeapShow is a powerful desk accessory which allows the user to view and manipulate the inner workings of practically any program, as it runs, on any size Macintosh or XL.

The major Macintosh developers are now using HeapShow to help light the way in creating new software and to diagnose problems in their existing programs.

Critic's Choice

"HeapShow can teach anyone - not just programmers - a lot about how the Mac actually works.
"This program should certainly be in all developers' toolkits. The rest of us can also get a lot out of it."

—Steven Bobker - MacUser

"For those of you who are heavily into programming on your Mac, a new utility, HeapShow . . . looks like a very useful development tool."

—Ebbe B. Ebbesen - The MACazine

"I can recommend a friendlier debugging tool . . . HeapShow . . .
"It follows Macintosh user interface conventions, and gives an information-packed display, so it is much easier to get started with . . ."

—Eric Zocher - The MACazine

- Includes Built-in Screen Protection Feature.
- Not Copy Protected.

B/T Computing Corp.
Linking Mind and Machine

P.O. Box 1465 Euless, TX 76039

30581

HeapShow™

for Graphic Software Analysis

VISION
inside
the Mind of Macintosh™

(Version 3.0)

Credits

Software: James C. Frazier
User's Manual: David & James C. Frazier

Thanks to Joan.

Second Edition, January, 1986
Copyright(C) 1985, 1986 by B/T Computing Corp.
All Rights Reserved.

B/T Computing Corporation
P.O. Box 1465
Euless, TX 76039

Customer Service

If you experience any problems with this software or its documentation, please call our customer service number.

Customer Service Phone Number
(817) 267-1415 Weekdays
9:00am to 5:00pm Central

Table of Contents

Appendixes

Introduction

When an application is running on the Macintosh, most of the action takes place in areas of memory called Heaps. With other performances such as cinema, theater, concerts or circus acts, what the audience ultimately sees on the screen or stage is only the end product of a much larger unseen effort, just as words spoken or actions taken are the result of the untold mental processes which preceded them. The quality of films, plays, and of a person's words and deeds is largely determined by the effectiveness and cooperation of what happens 'behind the scenes'. HeapShow has been designed to take users 'backstage' for a closer look.

If you could 'look' at the logic inside Macintosh while an application performs, you would find all the trappings of a live performance, most of which are located within the heap reserved for the application. Using props such as FONTs, ICONs and CURSors, supported by MENUs and WINDows, assisted by desk accessory DRVRs and PACKaged routines, the Mac's processor juggles STRings, draws PICTures, and performs practically any task imaginable according to CODEd scripts. All of these components exist as individual blocks of memory in Macintosh's RAM, and there can be hundreds or thousands of them in the heap at any one time, each performing its specific task. As an application is exercised, various types of these blocks are created, filled, moved, changed, emptied, linked, and regrouped as necessary to accomplish the user's requests. With all of this activity, the heap can become a very busy place.

But 'seeing' logic is not an easy thing to do. To human eyes, it is of course invisible; consisting of myriad tiny electrical charges going on and off within millions of RAM circuits millions of times per second. Advanced users and programmers who need to know what the logic is doing use utilities called debuggers to try to visualize the action. But these tools are basically stop-action microscopes. They are useful for examining the most minute of events, stepping through an intricate instant frame by frame (usually to find out what just caused the processor to fall off the stage).

Using a debugger to attempt an overall perspective of an average routine is like examining a stage crew molecule by molecule to determine its efficiency.

The other widely used hacker's tool is the disassembler, which provides a more or less complete listing of the individual machine instructions that make up the program. While definitely serving certain needs, studying an application by its disassembly listing is like trying to interpret an opera armed only with the score.

HeapShow fills many of the voids these tools were not designed to address. It provides graphic views of applications on much larger scales, including the important ability to see 'the whole picture' at once. Instead of dissection, HeapShow allows dynamic observation of the living subject, its habits, needs, strengths and weaknesses. And while few care to memorize endless sequences of machine instructions, HeapShow's graphic images and action sequences are readily remembered and more easily explained. A microscope is fine, as long as one knows where to start looking. HeapShow allows the user to pick targets from the overall picture, and then selectively zoom in for a closer look. When searching for a particular block or counting those of a certain kind, or whenever scanning the entire heap is required, no other tool matches the combination of HeapShow's graphic display coupled with the human eye. In short, HeapShow provides Vision into the Mind of Macintosh.

About This Manual

This manual has been organized to provide the assistance necessary to become familiar with HeapShow, its use, and its capabilities. We have attempted to arrange and present the information in a manner that will be of the greatest benefit to the broadest cross-section of our anticipated users.

The first section of this manual, Getting Started, describes procedures for loading HeapShow into your System files, and explains the need for two variants of the HeapShow driver.

A detailed description of HeapShow's display and the options and commands available to the user follows in Using HeapShow. A brief tutorial is included as an introduction to first-time users and to demonstrate HeapShow's newest features to users of older versions.

The section entitled Hazards of the Heap hilites a few common software design deficiencies that HeapShow can graphically point out, assisting developers in producing clean, efficient software products for the Macintosh.

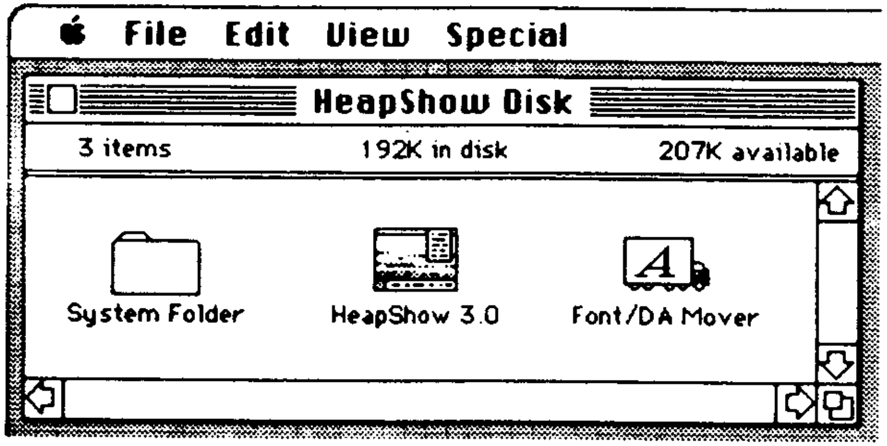
For the adventurous hacker, the section on Customizing HeapShow describes how to attach your own routines to HeapShow to handle that one essential task we didn't think of. Example source code fragments in 68000 assembly and Pascal are included to illustrate the process.

Appendix A provides a glimpse of the future by mentioning some of the upcoming enhancements to HeapShow.

Appendix B contains a personal note which might be of interest to you.

Appendix C is a miniature glossary containing a few simple definitions for some of the more technical terms used in this manual.

Finally, the Quick Reference in Appendix D is provided as a convenience to assist you when using HeapShow.



Getting Started

The HeapShow disk is not copy-protected and backup copies can be made using the Finder or any disk copying utility (consult your Macintosh owner's manual if this procedure is unfamiliar to you).

HeapShow is not an application program, but rather is in the form of a desk accessory (actually two accessories, described below). As with any desk accessory, HeapShow must be installed in an application's system folder before it can be used with the application.

The HeapShow disk contains a system file with both HeapShow drivers already installed. In addition, the disk has Apple's Desk Accessory Mover which can be used to install HeapShow into your own system folders. There are basically three ways to install HeapShow, depending on your unique requirements.

1) The easiest way is to simply replace the old System Folder on the desired disk with the new System Folder from the HeapShow disk. This will work fine in most cases, and has the advantage of updating the Finder and other system files at the same time. However, if your application contains a non-standard System Folder (often the case with copy-protected applications), or if you have made custom modifications to it, the new System Folder might not work.

2) If you want (or need) to keep your application's System Folder intact, you can use the included Desk Accessory Mover to move the HeapShow drivers into it. This will add HeapShow to the list of existing accessories without disturbing the rest of the System Folder's contents. (We recommend using a backup copy, just in case.) To add HeapShow to a System file on another disk:

* Insert the HeapShow disk into the built-in drive and the disk containing the application to which you wish to add HeapShow in the external drive.

* Open F /Desk Accessory Mover (double-click its icon).

* Click the Desk Accessory Mover button. The list on the left names all desk accessories in the System file on the HeapShow disk. The list on the right includes the desk accessories in the System file on your application disk.

* Select HeapShowS. (If both drivers are selected and copied at one time, the Desk Accessory Mover sometimes alters HeapShowS such that it loads into the application heap instead of the system heap.)

* Click the Copy button.

* Select HeapShowA.

* Click the Copy button.

* Click the Quit button to return to the Finder.

HeapShow will now appear in your application's Apple menu as two new desk accessories ready to use from within your application.

Note: You should not attempt to move any accessories into the startup System file (the one that is on the same disk as the currently running Desk Accessory Mover). If you do, the System file can be damaged with various (bad) results.

3) Finally, HeapShow can also be installed using a utility such as Apple's Resource Mover. A separate resource file containing both versions of HeapShow is included for this purpose. It is the file bearing the HeapShow icon entitled 'HeapShow 3.0', and can be discarded if you do not intend to use the Resource Mover.

wo Heaps -- Two Variations

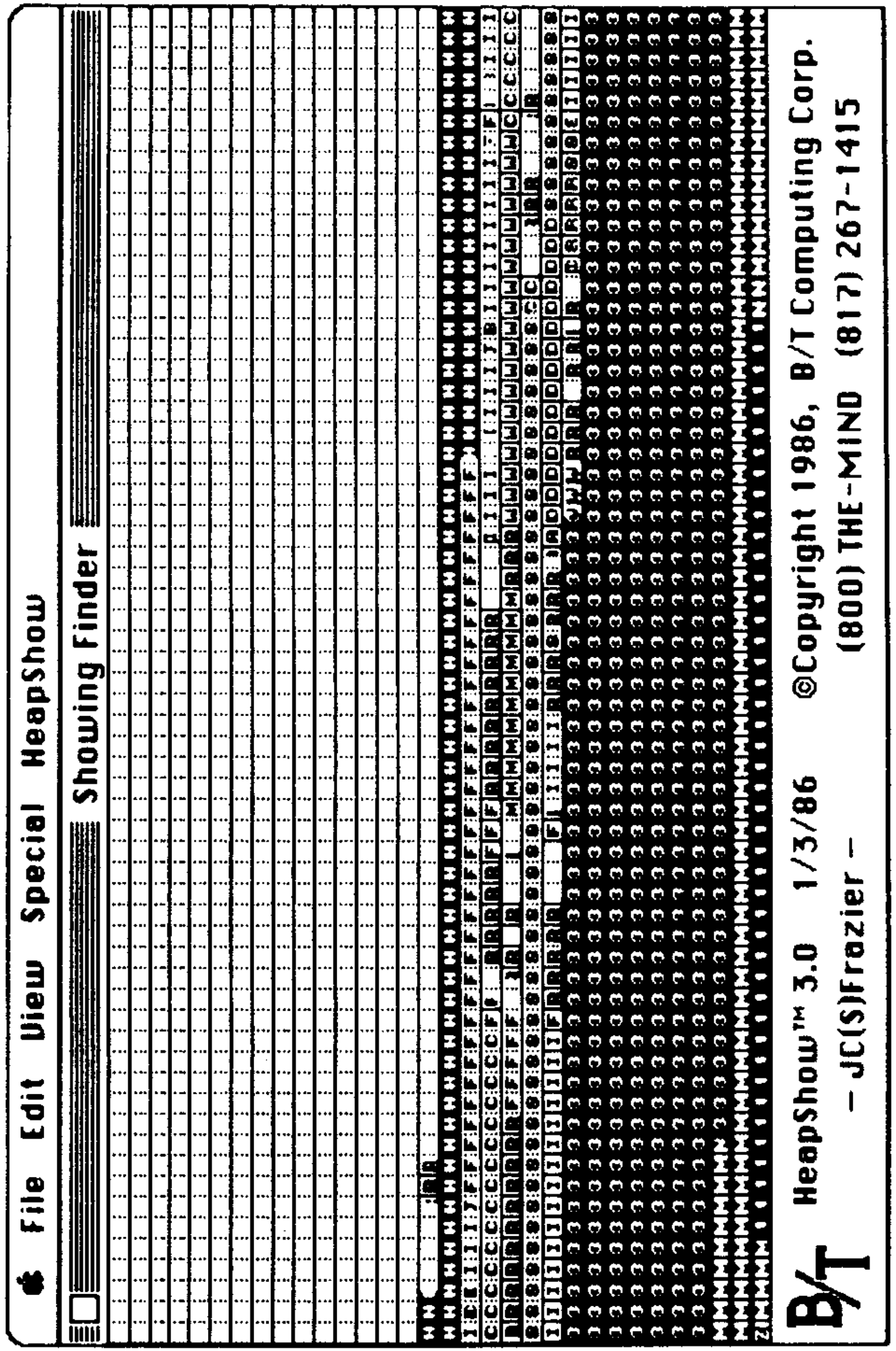
The HeapShow disk includes two variations of the HeapShow utility; suffixed A and S. The only significant difference between the two is that the 'A' version is loaded into the application heap when called, while the 'S' version operates from the system heap. Normally, the user will want to use HeapShowS so as not to disturb the application heap he or she is trying to study. By loading HeapShow into the less-active system heap, the application heap is left undisturbed for more accurate viewing.

Either version may be omitted from a target system folder if the user desires to conserve disk space (about 6K each). Each has the ability to examine either heap by toggling the appropriate menu item. The use and operation of the two versions are identical. However, since loading a desk accessory into the system heap is not a standard practice, there are a few extra considerations regarding the use of HeapShowS:

1) There just isn't enough room in the system heap of a 128K Macintosh for HeapShowS to be loaded. Use HeapShowA instead.

2) It is possible (though unlikely) for a situation to arise where HeapShowS gets loaded into the system heap, but later, it becomes necessary for the system to allocate the memory occupied by HeapShow. In this case, the system might bomb. If this happens to a given application, then using HeapShowA would be preferred.

3) Finally, because the system heap is not re-initialized when an application is terminated (as is the application heap), you should close HeapShowS before quitting. If HeapShowS is left open when you exit an application, the Finder could bomb on the way out. Nothing will be damaged, but it can be annoying.



HeapShow™ 3.0 1/3/86 ©Copyright 1986, B/T Computing Corp.
– JC(S)Frazier – (800) THE-MIND (817) 267-1415

Using HeapShow

Once you have copied the HeapShow drivers into your System file, you can run either utility. In operation, HeapShow will respond to various commands which may be given using HeapShow's menu, its control buttons, or the keyboard. The following tutorial sequence has been designed to quickly aquaint you with the power of HeapShow 3.0. We invite you to follow along, experimenting with any of HeapShow's options as we go. (If the screen happens to go dark while you are reading, don't worry, gently nudging the mouse will wake it back up. We'll describe this feature a little later.)

* Open an application you would like to study, or just use the Finder. Pull down the accessory (Apple) menu and select HeapShowA. (HeapShowS operates identically but for the extra considerations explained above.)

* HeapShow's window should now be visible, covering most of the desktop. If it isn't, try to verify that the installation procedure was performed correctly, and perhaps try HeapShow with another application or on another disk. If problems with HeapShow persist, you can call our customer support number found on the inside cover of this manual and someone at B/T will be happy to assist you.

* HeapShow displays information in two basic forms. It relies heavily on its graphic display to present the maximum amount of information to the user in a manner that can be quickly interpreted and understood. Whenever possible, it supplements the graphic display with more specific information which is better represented in the form of text. As will be seen, the graphic display can be easily tailored to the user's needs. Textual information is either displayed automatically (e.g. the address indicated by the cursor), or in response to a user command (e.g. descriptive data about the currently selected block).

* A new‚y opened HeapShow window displays information about itself so that the user will know what version is being used, etc. The graphic display starts off showing the application heap, beginning with its HeapZone Header. (In lay terms, this is the area of memory containing the program being viewed, starting from its beginning. For advanced users, upwards from ApplZone.)

* Press the spacebar, or click the mouse in the text area at the bottom of the window. The version and copyright are replaced by a set of control buttons as well as information about the application heap, the program stack and the graphic display.

The Graphic Display

HeapShow graphically represents the contents of Macintosh's memory by drawing an image on the screen for each logical block, using distinctive patterns and symbols to indicate the block's type and status. The first block encountered (the one with the lowest address) is drawn from the lower-left corner of the display towards the right as far as necessary to indicate the block's size, using the current display scale. If an image reaches the right display border, it wraps up to the next higher row and continues, somewhat like an inverted word processor display. The first image is immediately followed by an image representing the next block in memory, and so on, until either the display is filled or the entire content of the heaps is represented. Any unfilled area at the top of the display is left white indicating that this region of memory lies beyond the top of the application heap.

* Using the mouse, move the cursor to the lower-left corner of the graphic display. The line labelled 'Cursor' displays the RAM address associated with the tip of the arrow in hexadecimal (to make it easy to use in conjunction with other programming/debugging tools and manuals).

* Slowly move the cursor horizontally to th right. The addresses steadily increase as you go. When you reach the right border, reposition the cursor at the left edge of the next higher row to continue up the heap.

* Now move the cursor vertically upwards at the same speed as before. The addresses now increase at a much more rapid rate since you are scanning row by row instead of pixel by pixel. When you reach the upper-right corner, you are over the highest RAM address being displayed, and must scroll the display to continue. This type of display was chosen as a more intuitive representation of memory, with lower addresses shown at the bottom and higher addresses toward the top.

Each block's image is filled with a pattern or symbol which identifies the block's type (e.g. FONTs are filled with the letter F, ICONs with the letter I, etc.) In addition, each block's mobility can be determined by its background pattern:

Solid Black -- Immobile (NonRelocatable or Locked)
Solid Frame -- Relocatable, but not Purgeable
Dashed Frame -- Purgeable as well as Relocatable

If the block contains a Macintosh resource, the block's symbol will be the first letter of its resource type, unless the resource is of type 'CODE'. Code segments contain the last digit of the segment number (Resource ID) to make them easier to distinguish.

Non-Resource blocks (such as Relocatable data, NonRelocatable Blocks, etc.) display an extra-large symbol which usually is the first letter of the block's type or kind.

NN -- NonRelocatable Block
WW -- Window Record (NonRelocatable)
MM -- Master Pointer Block (Never-Relocatable)
ZZ -- HeapZone Header (not really a block at all)
RR -- Relocatable (even if locked or purgeable)
⬚ -- Free Memory Block

Window Records

WWW	Normal Window Record (NonRelocatable)
WWW	Front Window's Window Record

HeapShow treats window records as a special case of non-relocatable memory, since they are quite often the focus of an application's attention. A bold W is used to distinguish them from other non-relocatable blocks. HeapShow further distinguishes the window record belonging to the front window by inverting its normal pattern (a bold black W on a white background). This feature can be used to locate and track the records of each window an application uses by noting which blocks light up at various times.

Buttons and Keys

Scroll Up 1 Row	↑	X	Magnify Display
Scroll Down 1 Row	↓	Z	Zoom Out
Toggle Display Height	↕	I	Invert Alternate Blocks
Toggle Display Width	↔	▣	Darken Screen Now

The HeapShow display contains a cluster of control buttons in the lower left portion of the text display area. Depressing the shift key selects the alternate cluster of buttons, which may be locked in place by depressing the capsLock key. For each button, there is also a keystroke alternative for those times when using the mouse is less convenient. Together, these controls give users greater flexibility over HeapShow's display.

Scrolling the Graphic Display

↑ ↓ **Keys: < > 1..0 A S ~**

The two scroll buttons are used to bring areas of memory above and below the displayed range into view. Scrolling upward moves the display into higher RAM addresses, moving toward the program stack. The highest address for which contents are displayed is the limit to which the application heap may grow (ApplLimit). Though higher addresses may be indicated by the cursor (up to a maximum exceeding four megabytes), the current version of HeapShow will not indicate their contents. Scrolling downward moves into lower addressed RAM, eventually displaying the system heap, just below the beginning of the application heap. The lowest address that the current version of HeapShow can reach is the system heapZone header (SysZone).

* Click the scroll up button. The graphic display scrolls up one row at a time. The amount of memory the row represents depends on the viewing scale, and, if no block is selected, is displayed in the text area of the window.

* Press the 1 key. This also causes the display to scroll one row. Pressing a key from 2 thru 0 will scroll from 2 to 10 rows for rapidly locating the area you wish to view. When scrolling by use of the keyboard, the direction of the scroll will be the same as that of the last scroll button hit.

* Press the '<' key. (HeapShow ignores case, so it isn't necessary to press the shift key.) This changes the scroll direction to downward without actually scrolling. Pressing a numeric key now will cause the display to scroll down that number of rows, unless stopped by reaching the beginning of the system heap. Pressing the '>' sets the scroll direction to upward.

Changing th_ Viewing Scale

Keys: X Z

* Click the 'X' button or press the 'X' key. This command reduces the viewing scale by one-half. Half as much of the heap is now displayed, but each block's image is now twice as large, making it easier to identify smaller components.

HeapShow opens at a viewing scale of 1:8. That is 1 bit (or pixel) of display represents 8 bits (1 byte) of the heap's RAM. Each use of the 'X' button or key reduces the viewing scale by half, until the minimum scale of 1:1 is reached, in which even the smallest possible blocks are clearly visible.

* Press the 'Z' key or click the 'Z' button. This 'Zoom-out' command has exactly the opposite effect of the 'X' command. Each use doubles the viewing scale, showing half as much detail, but twice as much heap.

* Hit the 'Z' button or key repeatedly until the maximum scale of 1:256 is reached (as indicated by the display line labelled, 'Scale'). As the text display indicates, the full-sized graphics display now represents four megabytes of RAM, thus showing in one screen the total heap contents of even the largest of the Monster Macs!

Resizing HeapShow's Window

Keys: H W F Q

* Depress the capsLock key while we exercise the alternate set of control buttons. The alternate cluster replaces the scroll/scale cluster anytime the shift or capslock keys are depressed. (Note that the keyboard alternatives for these buttons do not require the shift or capsLock keys to be depressed.)

HeapShow's window can be resized to facilitate its use with any other window(s) you might be using. However, due to the unique nature of the graphic display, a non-standard sizing mechanism is used, which limits the available sizes to any combination of full or half width, and full or half height.

* Click the 'Height' (upper) sizing button or press the 'H' key. This reduces the height of HeapShow's window by one-half, thus reducing the amount of desktop it covers. This can be especially useful for viewing programs that do not have horizontal scroll capabilities (e.g. many word processors).

* Click the 'Height' button again. The screen toggles back to its full size.

* Click the 'Width' button or press the 'W' key. The window is reduced to one-half of its full width. This is helpful when viewing programs without vertical scroll, or columnar displays (e.g. spreadsheets).

* To reduce both the window's height and its width, either use both the 'H' and the 'W' commands, or simply press the 'Q' key. If you have previously moved the window on the desktop, note that the 'Quarter Window' command restores its original location.

* Press the 'F' key to restore the original Full size and location.

Darkening the ᴗᴗreen

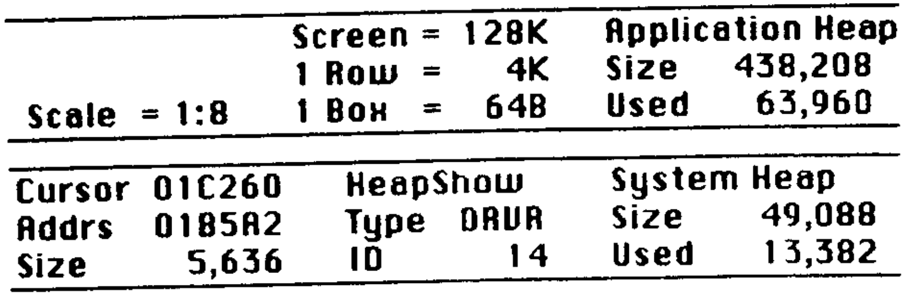
Key: D

* Click the button that looks like a little Macintosh or press the 'D' key. After a moment (to give you time to put the mouse aside), the screen goes completely dark without requiring you to adjust the brightness control knob. If you are using an XL, you finally have a way to COMPLETELY darken the screen when not in use.

* Move the mouse or press the shift, capsLock, option or command key. The display is completely restored (at the brightness level you had previously set). Actually, any event or condition the Macintosh can recognize will restore the screen, but since moving the mouse, or pressing modifier keys do not generate events, using them to wake Macintosh is safer (in case you have an application's window on top of HeapShow's when the screen is blanked). There are two 'quirks' to screen blanking: first, after the screen is restored, you may notice that a free block of about 21K (32K on an XL) has appeared in the middle of the application heap. This is the footprint left by a non-relocatable block HeapShow created in which to save the screen image. If there isn't room in the heap for such a block, HeapShow will still darken the screen, but will just redraw it instead of restoring it from its buffer.

The second 'quirk' is that ANY event will awaken the Mac. This may cause unusual behaviour in some cases. When printing, some applications are slow enough to allow the screen to darken automatically between operations (about 2 1/2 minutes), then generate an event which lights the screen back up. This :ycling may seem strange, but since HeapShow doesn't disturb ɪɪe events, it doesn't hurt anything.

The Text Display



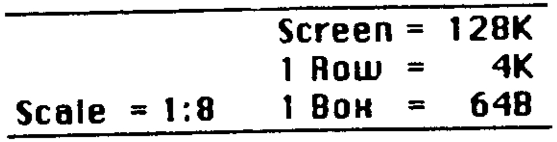
```
                  Screen =  128K   Application Heap
                  1 Row  =    4K   Size     438,208
Scale = 1:8       1 Box  =   64B   Used      63,960

Cursor  01C260    HeapShow         System Heap
Addrs   0185A2    Type   DRVR      Size      49,088
Size     5,636    ID       14      Used      13,382
```

The text display at the bottom of HeapShow's window is used to supplement the graphic representation. Each of the display lines are described below.

Cursor – as the cursor is moved around in the HeapShow window, this field designates the RAM address represented by the point at which the cursor is pointed.

Scale Information



```
               Screen =  128K
               1 Row  =    4K
Scale = 1:8    1 Box  =   64B
```

Scale – the current viewing scale in which HeapShow is representing memory, measured in pixels to memory bits, or screen bytes to RAM bytes.

Screen – the total number of memory bytes represented by the entire HeapShow graphic screen.

1 Row – the total number of memory bytes currently represented by each row in the display.

1 Box – the total number of memory bytes currently represented by each 8 x 8 pixel box in the display.

Heap ..formation

Application Heap	
Size	438,208
Used	64,164

System Heap	
Size	49,088
Used	13,382

Application/System Heap - This label indicates which heap is active and is important since the HeapShow options to Compact the Heap or Maximize FreeMem apply only to the 'active' heap whose label is shown, even though either or both heaps may be showing on the graphic display.

Size - the total number of bytes in the active heap.

Used - the total number of bytes in the active heap that are not labeled as free memory.

Stack Information

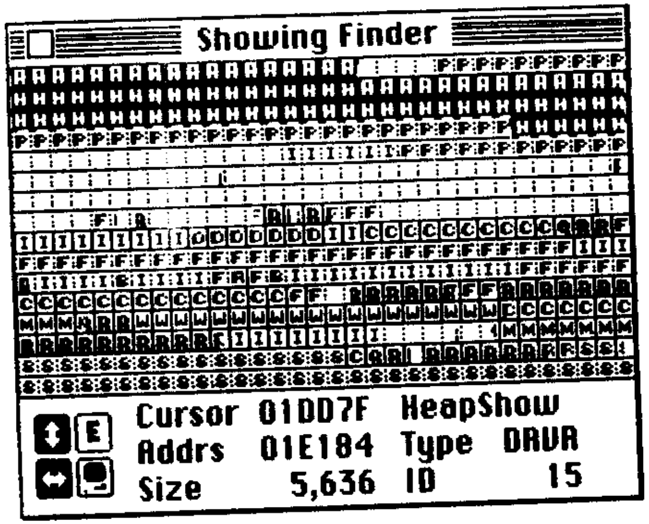
Stack	079B14
Size	8,354
Used	180

Stack - the address (in hexadecimal) of the stack pointer (A7) upon entry into HeapShow. Not particularly useful, but fun to watch on a runaway application.

Size - the maximum size in bytes to which the stack may grow without colliding with the top of the application heap. (StackBase - ApplLimit)

Used - the number of bytes currently in use by the stack. (StackBase - SP)

Selected Block Information



Cursor 01DD7F HeapShow
Addrs 01E184 Type DRVR
Size 5,636 ID 15

* Click on the image of any block represented by the graphic display. The border of the selected block becomes hilited and additional information about the block replaces the Scale information.

Address - the logical RAM address of the selected block, in hexadecimal. Note that every block in the heap is preceded by an eight-byte block header. Former versions of HeapShow displayed this PHYSICAL address, but we have found the LOGICAL address of the block's contents to be more useful. (The graphic display still represents the physical size of each block since that area of RAM is nonetheless occupied.)

Size - gives you the logical size of the selected block in bytes. Note that in addition to the block header (see above), blocks may have additional unused bytes at the end. The total 'overhead' or difference between a block's logical size (its contents) and its physical size (the space it takes up in RAM) can vary between 8 and 22 bytes, but is unimportant in most cases.

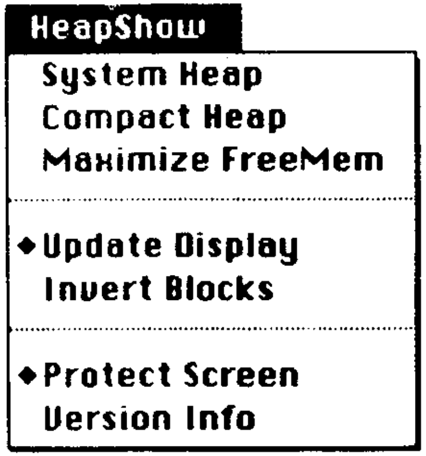
Name - the second column of text information will begin with the block's resource name, if it has one. (The serious user can use a resource-editing utility such as Redit or RMover to name resources in the system and application files. HeapShow will thereafter display these names when the resource blocks are selected. It is not necessary to name the individual font resources since HeapShow can figure them out.)

Type - the selected block's resource type is shown if the block contains Macintosh resource. Otherwise the label will indicate the kind of block.

ID - if the block contains a resource, its resource ID number is displayed.

* Clicking anywhere in the text display area will deselect any previously selected block, and restore the viewing scale information.

HeapShow's Menu



The items on HeapShow's menu are accessed in the standard Macintosh manner, except that any item may be alternatively selected by pressing the key which corresponds to the first letter of the item (using the command key is not necessary). The first three items are toggles, switching back and forth between alternate modes, with diamonds indicating whether a particular feature is on or off.

Application/System Heap

* Select the 'System Heap' option from the HeapShow menu (or press the 'S' key). The graphic display now begins with the system heap's zone header, and information about the system heap is displayed in the text display area. If you choose the 'Compact Heap' or 'Maximize FreeMem' commands from the menu, they would now affect the system heap since it is presently the active heap.

* Selecting the same item (which now shows 'Application'), reselects and redisplays the application heap. Pressing the 'A' key has the same effect.

Compact Heap

Selecting this item will cause a compaction of the active heap without purging any blocks. This moves relocatable blocks as close as possible to the beginning of the heap (unless the heap is fragmented -- see the section on Hazards below), joining the individual free blocks together into one or more larger free blocks for greater memory efficiency. This is particularly important in cases where the application needs to allocate a single block larger than any of the individual free blocks.

Maximize Free Memory

This item causes a combination of purging and compaction to occur, generally resulting in the optimum amount of free memory and leaving the heap in a state of maximum efficiency. It purges all purgeable blocks from the displayed heap, thus freeing the memory they occupied. This often improves the overall performance of the application being viewed (sometimes dramatically), and can even prevent out-of-memory bombs.

Update Display

Normally, HeapShow updates its display every 4 seconds to give the user a dynamic view of the inner workings of the Mac in operation. Unfortunately, even though great effort has been made to make the update as fast as possible, it still requires a certain amount of time to complete, and thus tends to slow things down a little. If this delay is objectionable, the automatic update mechanism may be disabled.

Invert Blocks

Whenever two or more contiguous blocks of the same type occur in memory, it can be difficult to determine the boundaries between them. Selecting this item (or pressing the 'I' key) causes HeapShow to invert the image of every other block, making the boundaries between them stand out clearly.

Screen Protection

Users should be aware that when the Macintosh screen is left at its normal viewing brightness for long periods of time, "screen burn-in" will occur which results in a permanent image of the displayed material on the screen. This can happen when users forget to darken the screen before leaving the Mac unattended, but much more often it occurs right in front of the user while he or she isn't looking. In the middle of entering some source code or figuring your taxes, you need to look something up. You don't notice the time escaping, but as your faithful Mac waits patiently for you to resume, the constant bombardment of photons is carving a permanent negative of the menu bar. Maybe only ten or fifteen minutes go by, but how often does this happen? Radiation exposure has a cumulative effect, (and by the way, not all of those nasty little photons stop on Mac's side of the screen...)

This next step is about the easiest one you'll ever have to learn.

* Do nothing. (For about 2 1/2 minutes.) Unless one of your experiments was to turn off the automatic screen protection (using this menu item or the 'P' key), your Mac should by now have gratefully shut his 'eyelids', thus protecting his screen and your face from needless 'burn-in'. When you're ready to resume, just move the mouse or tap a key (it's safer to use the modifier keys since they don't generate events).

This menu selection allows users to turn off the automatic protection if they so desire. Note that HeapShow does not have to be the active window, for the screen protection to work. It only needs to be open somewhere on the desktop. Unfortunately, some applications refuse to operate with desk accessories open (e.g. MacPaint and MacDraw), and thus deny you the benefit of this protection while using them. If you are unsure whether HeapShow's protection works with any particular application, it would be wise to test it before leaving a bright screen unattended.

Hazards of the Heap

HeapShow provides a graphic representation of the dynamic conditions within the Macintosh's memory, as the program is operated and tested. Often, a single glance at HeapShow's display will make obvious what hours of debugging would fail to uncover. Below are a few software design deficiencies that HeapShow can point out to assist in the development of efficient applications.

Leftovers

Resources, code segments and other blocks of memory loaded into the heap when needed, but then left there after use (often in an unpurgeable state). A code segment which is never locked is a good indication of such a problem. While Macintosh's built-in Memory Manager will often purge unnecessary blocks as it needs them (this is not always true), just having the purgeable blocks floating around in the heap increases the load on the system. And many 'virtual' systems would rather go to disk (SLOW!) than force a purging of the heap.

Cross-calling

The unintentional placement of small, frequently accessed routines in large, seldom used segments of code, requiring the constant reloading or permanent residence of unnecessary bulk. HeapShow demonstrates cross-calling by the brief but frequent flashing of code segments as they are locked, accessed and unlocked in rapid succession.

Fragmentation

As described in the 'Inside Macintosh' documentation, this occurs when small islands of immobile blocks are located far out in the heap in such a way that they interfere with the consolidation of free memory through compaction. This problem is easy to spot with HeapShow, showing up as isolated blocks with black backgrounds well out in the heap. The problem becomes especially acute when the fragmenting blocks are master pointers, because they cannot be relocated or removed until the application is terminated.

Danglers

A very common problem with any developing application (and a few finished ones), is that of the dangling handle. Sometimes the block to which a handle refers has moved, but the code blindly sends data to the old address. In other cases, the needed block has been purged, but the application fails to notice its absence until it is too late. These situations are often lethal, but can be particularly difficult to detect, since moving or purging the block in question might only occur after some specific chain of events. HeapShow's abilities to Compact the heap and to Maximize FreeMem can be used by developers to force a radical rearrangement of the blocks in memory, quickly bringing problems of this type out into the open.

Helpful Hints

* Before installing HeapShow into any System files, it's always safer to first make a backup of the HeapShow disk and of each of the application disks into which you will be installing HeapShow.

* For advanced users -- If Apple's Font/DA Mover is giving you trouble installing HeapShowS such that it loads into the application heap, using RMover or REdit to set its system heap loading bit after copying the DRVR into your system file can cure the problem.

* In addition to using the numeric keys to scroll more quickly through large heaps, long distances can be covered rapidly by increasing the viewing scale before scrolling.

* If you would like hard copies of what you see in HeapShow's displays, using command-shift-3 will generate a MacPaint picture of the screen, and command-shift-4 will send the screen image to a printer.

* With applications which work properly with open desk accessories, opening HeapShow as soon as you open the application gives the user the benefit of screen burn-in protection from the start.

* XL owners are aware that the Lisa screen doesn't get very dim, and that there is no manual control as on the Mac. Opening HeapShow just before leaving the XL allows the screen to be completely blanked when not in use.

* HeapShow can be kept around on the desktop, ready to blank out the screen quickly, in case you don't want others to see what you're really doing instead of what you're supposed to be doing.

* Using HeapShow with Switcher can be really interesting. If you'll open HeapShowS once and open HeapShowA in each of the applications you're using, you can switch around, seeing the same RAM from different points of view.

* HeapShow can help you determine how many times to call MoreMasters at the beginning of the application you are writing. Temporarily remove all MoreMasters calls from your code and recompile it. Run your application and open up all of the windows, desk accessories, utility routines and data blocks that anyone using your program is likely to open. Then use HeapShow to see how many blocks of Master Pointers it took to handle the load. Adding a 20-30% margin for safety will give you a good estimate of the number of calls you should make to MoreMasters.

* For developers using debuggers like MacDB, half the time is spent just searching memory, trying to find the place to start looking for a problem. Running HeapShow on the target machine can save hours. Just before interrupting the Mac, select the code segment or other block you assume is at fault, and if you have some idea about how far into the block you want to start your search (e.g. half way, one third, etc.), point the cursor at the suspected area. Once inside the debugger, you can enter the block's starting address, or the address you were pointing to, both of which will be shown on HeapShow's now frozen display.

* So far, the two stangest applications we have looked into are Jazz, which keeps several code segments hovering at the top of the application heap, bringing them down low only when in use, and Excel, which sets up huge data buffers in the oddest places. (The dullest application to watch seems to be MacPaint, which doesn't appear to do much of anything.)

Customizi _ HeapShow

In order to provide greater flexibility to the user, HeapShow has been designed to allow the attachment of a custom routine to its update mechanism. While this is primarily intended for the more advanced user or developer, the process is fairly simple. (Example source code fragments are provided in 68000 assembly and Lisa Pascal to demonstrate the process.)

First, write a routine (the hard part), load it into memory in which ever way you desire and lock it down. Determine the start address of the routine in memory and place the address in the DataHandle field of HeapShow's window record.

The custom routine may be written in any language and in whatever form the user desires. Of course, it must be assembled or compiled down to machine code before attaching to HeapShow.

The routine can be loaded as a code segment, as a separate resource, or as another desk accessory.

Determining the start address of your routine need not be difficult, but caution is advised. Most resource types have some header information at the beginning, so that just passing its handle (de-referenced to a pointer) to HeapShow will not work. See Apple's 'Inside Macintosh' documentation for specific details about the type of resource you intend to use.

The routine must be locked down in memory unless you intend to provide some mechanism for constantly updating the pointer in DataHandle. Otherwise, HeapShow could JSR to the old location with disasterous results (the bomb!).

Each time HeapShow needs to update its display, it first checks the DataHandle field for a non-zero value. If it finds one, before handling its screen update, it performs a JSR to the address it found. (NOTE: DO NOT PLACE ANY OTHER VALUE IN THIS FIELD OR HEAPSHOW WILL JSR ITS BRAINS OUT!) Your routine will thereafter be called every time HeapShow updates its display (normally every four seconds). To turn off your custom routine, simply restore the DataHandle field to nil (O).

Example Code Fragments for Linking Custom Routines

The following code fragments are provided as examples of linking custom user routines to HeapShow's update.

68000 Assembly Code Fragment

```
;-----------------------------------------------------
;This example demonstrates a method of linking where
;the routine is attached to the user's main program.

HeapName
        DC.B    9               ;length of HeapShow's name
        DC.B    'HeapShowA'     ;name of HeapShow accessory

InitHeapLink

;Call this routine once from your main program to link your
;custom routine's address into the wDataHandle field of
;HeapShow's window.

        CLR.W   -(SP)           ;make room for refnum result
        PEA     HeapName        ;pointer to HeapShow's name
        _OpenDeskAcc            ;call to open HeapShow
        MOVE.W  (SP)+,D0        ;pull refnum off stack

;Since HeapShow owns the most recently opened window,
;we don't have to scan through the window list to find it.
;If we did, we could compare the refnum we just pulled
;from the stack to each window's windowKind field.
;HeapShow's will match.

        MOVE.L  WindowList,A0       ;pointer to frontwindow
        LEA     MyRoutine,A1        ;get address of entry
                                    ;of the custom routine
        MOVE.L  A1,wDataHandle(A0)  ;and make the link
        RTS                         ;return to main program

;'MyRoutine' will now be called by HeapShow each time it
;updates its display (normally about every 4 seconds).
;-----------------------------------------------------
```

```
;To disconnect your routine, simply replace the pointer
;in the wDataHandle field with nil (0000).

UnlinkHeapShow

;This simple routine assumes that HeapShow's window is still
;up front.  You might want to add some code to verify this,
;or else loop through the windowlist to find HeapShow's.

        MOVE.L  WindowList,A0   ;point to HeapShow's window
        CLR.L   wDataHandle(A0) ;and zap the link
        RTS                     ;return to main program

;HeapShow will no longer call your routine until the link
;is re-established.

;---------------------------------------------------
MyRoutine

;This example just beeps the speaker each time it is called.
;It demonstrates how simple a custom routine can be.

        MOVE.W  #8,-(SP)        ;push count for the beep
        _SysBeep                ;and beep it
        RTS                     ;return to HeapShow

;It is not necessary to save and restore the registers
;since HeapShow saves all the working regs (A1-A4/D1-D7)
;before JSRing to your routine.  Of course nothing should
;be done to HeapShow (like closing it) while in your
;routine, or else you might return to the Twilight Zone.
```

Pascal Source Code Fragment

This example demonstrates a method of linking where the custom routine is part of the user's main program.

```
{-----------------------------------------------}
procedure InitHeapLink;

{Call this routine once from your main program to link
 your routine to the HeapShow window's DataHandle field.}

const
  HeapName = 'HeapShowA';              {name of acc to open}

var
 tempwindow : windowpeek;    {needed to access DataHandle}

begin
 refnum := OpenDeskAcc(HeapName);           {open HeapShow}
 tempwindow := windowpeek(frontwindow);        {its window}

{Since HeapShow owns the most recently opened window, we
  don't have to scan through the window list to find it
  If we did, we could compare refnum to each window's
 windowKind field.  HeapShow's will match.}

 tempwindow.DataHandle := @myRoutine; {link routine address}
 end;                                      {all done linking}

{Your routine 'MyRoutine' will now be called by HeapShow
 each time it updates its display.}

{-----------------------------------------------}
```

```
{To disconnect your routine, simply replace the pointer
 in the DataHandle field with nil.}

procedure UnlinkHeapShow;

{This simple routine assumes that HeapShow's window is still
 up front.  You might want to add some code to verify this,
 or loop through the windowlist to find HeapShow's window.}

var
  tempwindow : windowpeek;          {to access DataHandle}

begin
  tempwindow := windowpeek(frontwindow);        {its window}
  tempwindow.DataHandle := nil;             {unlink routine}
  end;                                  {all done unlinking}

{HeapShow will no longer call your routine until the link
 is reestablished.}
{----------------------------------------------------------}
procedure MyRoutine;

{This example just beeps the speaker each time it is called.
 It shows how simple the form of a custom routine can be.}

begin
  SysBeep(8);                       {beep for 8 tics}
  end;          {routine is done, return to HeapShow}

{----------------------------------------------------------}
```

Appendix A. Future Enhancements

The following is a partial list of enhancements and additions to HeapShow that are likely to be made within the coming months:

* The range of memory displayed by HeapShow will be expanded to include non-heap areas such as the stack, global variable areas, and the 'voids' between them. [Version 4.0]

* Options for hex/ascii, graphic and string dumps (e.g. icons, cursors, pictures, patterns, etc.). [Version 4.0]

* The ability to edit the contents of memory while an application is running. [Version 4.0]

* The ability to post events to an application to test their effect, or to allow macro control. [Version 4.0]

* The ability to selectively manipulate blocks in the heap (e.g. create, load, lock, unlock, release, copy, move, resize, set purgeability, alter contents, etc.). [Version 4.0]

* The ability to generate statistical information about an application in operation.

* A disassembler (and possibly reassembler) will be linked to HeapShow to allow the decoding, patching, etc. of code segments in memory (which could then be saved to disk with a utility such as DiskShow).

* The ability to save information about various structures in memory for delayed replay or analysis.

* Run-time error trapping and handling to allow application recovery from many system errors, data recovery from all but the worst.

Appendi : A Personal Note to You

We at B/T would like to ask you, the user, to make one consideration on our behalf. It has taken a lot of time and hard work to put HeapShow together. It has taken a lot more to draft and edit this manual, and to design and assemble the package. And it will undoubtedly take even more to bring the fruit of our labors to the public, as well as to provide you the kind of continuous, long-term support that you might need from time to time. The future enhancements and upgrades will demand even more precious time and effort.

As with any business, a large part of our incentive to spend our resources on the development of such tools as HeapShow and DiskShow has to be the financial return generated by their distribution. For this reason, it is essential that you not engage in your own private distribution of the products we provide to you. Each free copy you give to a friend (believe us, we know how persuasive friends can be!) deprives us of a portion of those sorely needed revenues. If each HeapShow user gives away just one copy, the revenue we receive for our effort is not just cut by half, it is LITERALLY DECIMATED, since the friend you give a copy to will likely also give a copy to a friend, who will give a copy to a friend...

Therefore, if you like HeapShow, you should encourage the continued creation and development of such tools, and help support our efforts to support your efforts. Don't give out copies of our products, give out our name and number so your friends can buy (and register) their own copy.

And if you have been the recipient of a gratuitous copy of HeapShow, then by all means take advantage of the opportunity to try it out. If you don't like it, don't use it (though we would still appreciate hearing your comments). But if you do use it, order your own personal copy so the time and money we spent developing HeapShow can be repaid. We may be doing the same for you someday!

We sincerely hope you enjoy HeapShow!

Rick, Dave, Scott & Joanie of B/T Computing

Appendix C: Glossary

Some of the terminology presented in this manual may be unfamiliar to the inexperienced user. We have chosen to define a few of the more basic terms so that the fundamental concepts described in this manual can be understood. More detailed information can be obtained from Apple's 'Inside Macintosh' which is a worthwhile investment for those with more serious intentions.

* Allocate - to reserve an area of memory for use.

* Application Heap - the portion of the heap available to the running application program for its own allocation and use.

* Block - an area of contiguous memory on the heap.

* Compaction - the process of moving allocated blocks together within the heap in order to collect the free space into a single block.

* Heap - the area of memory in which space is dynamically allocated and released on demand, using the Memory Manager.

* Master pointer - a single pointer to a relocatable block, maintained by the Memory Manager and updated whenever the block is moved, purged, or reallocated.

* Purge - to remove a relocatable block from the heap, leaving its master pointer allocated but set to NIL (not pointing to anything).

* Relocatable block - a block that can be moved within the heap during compaction.

* Stack - the area of memory in which space is allocated and released in LIFO (last-in-first-out) order.

* System Heap - the portion of the heap reserved for use by the Toolbox and Operating System.

Keyboard Command Reference

For added convenience, nearly all of the operations and commands of HeapShow can be performed with a keystroke as an alternative to using the mouse. These options are listed below. (HeapShow is not sensitive to case.)

Key	Action
A	Application heap is selected and displayed
S	System heap is selected and displayed
1..0	Scroll the display from one to ten rows at a time
<	Set the scroll direction to Down
>	Set the scroll direction to Up
~	Reset the display and scroll direction
X	Magnify the viewing scale for a closer look
Z	Zoom the viewing scale out to see more data
H	Height of the display window is toggled
W	Width is toggled
Q	Quarter size display window
F	Full size display window
D	Darken the Macintosh screen
C	Compact the selected heap
M	Maximize the free memory in the selected heap
I	Invert alternate blocks for better distinction
U	Updates to the display are suspended/resumed
P	Protection from burn-in is suspended/resumed
V	Version information is displayed
Cmd-Shift-3	Save display as MacPaint document
Cmd-Shift-4	Send display to imagewriter

IMPORTANT NOTE

You are licensed to use this product only if you agree to the terms of this agreement. You are not authorized to use the software described by this documentation until you have read the following license agreement, completed and signed the Registration Card, and returned it to B/T Computing Corp. We rely heavily on your honesty and integrity in refraining from the unauthorized use, copying or distribution of this software.

END USER LICENSE AGREEMENT

B/T Computing Corp. grants the purchaser of this software product a non-exclusive license to make personal use of the software, subject to the terms and restrictions of this End User License Agreement. B/T retains title to all such software.

The enclosed software and documentation are copyrighted. You are not authorized to transfer the software electronically or distribute copies of the software or documentation to others, whether or not you demand or receive compensation. This software may not legally be modified or translated without prior written consent from B/T, however such consent will normally be granted in cases where the user desires the modifications strictly for his or her own personal use. With proper agreements, the source code for this software product may be made available to the end user for such purposes.

WARRANTY AND LIABILITY LIMITATIONS

If a defect in the disk on which the software described herein is provided occurs within 90 days of purchase by the original purchaser of this product, the disk may be returned to B/T or its authorized dealer, along with dated proof of purchase, for replacement without charge. This is your sole remedy in the event of a defective disk

The software (due to its very nature) is licensed and provided 'as is', without warranty of any kind other than the expressed warranty set forth above.
B/T MAKES NO OTHER WARRANTIES, GUARANTEE OR REPRESENTATIONS, EITHER EXPRESSED OR IMPLIED, BY STATUTE OR OTHERWISE, REGARDING THE SOFTWARE PRODUCT, ITS QUALITY, ITS PERFORMANCE, ITS MERCHANTABILITY, OR ITS FITNESS FOR A PARTICULAR PURPOSE.
The liability of B/T under the warranty set forth above shall be limited in any case to the amount paid by the purchaser for the product. In no event shall B/T be liable for direct, indirect, incidental or consequential damages resulting from the use, the results of use, or the inability to use the software.

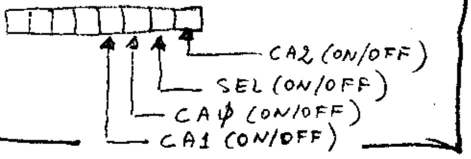
The warranty information set forth above shall be governed by and construed according to the laws of the State of Texas.

Sony Driver

```
>PC 4195  05C4
>TD
 05C4:                     BSR.S    *-$0018              ; 004185AC
Pc=004185C4 SR=0000A000 TM=00002EDA
D0=00000008 D1=0000017C D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
A4=0000CC28 A5=000F7634 A6=000F68BC A7=000F6852
>T
4185AC:                    MOVEA.L $01E0,A0  <- IWM = dBase
PC=004185AC SR=0000A000 TM=00002EDB
D0=00000008 D1=0000017C D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
A4=0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
>
4185B0:                    MOVEA.L $01D4,A2  <- VIA  = vBase
PC=004185B0 SR=0000A000 TM=00002EDC
D0=00000008 D1=0000017C D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
A4=0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
>
4185B4:                    MOVEA.L $0134,A1  <- T_SonyPtr
PC=004185B4 SR=0000A000 TM=00002EDD
D0=00000008 D1=0000017C D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
 =0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
>
4185B8:                    MOVE.W  $0012(A1),D1 <-
PC=004185B8 SR=0000A000 TM=00002EDE
D0=00000008 D1=0000017C D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
A4=0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
>
4185BC:                    ADD.W   D1,D1              (*2)
PC=004185BC SR=0000A000 TM=00002EDF
D0=00000008 D1=00000006 D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
A4=0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
>
4185BE:                    MOVE.W  *-$0022(D1.W),D1 ; 0041859C + φC
PC=004185BE SR=0000A000 TM=00002EE0
D0=00000008 D1=0000000C D2=A0000003 D3=434F4445
D4=00000000 D5=00000005 D6=0000011C D7=00000000
A0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
A4=0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
>
4185C2:                    RTS
C=004185C2 SR=0000A000 TM=00002EE1
 =00000008 D1=0000017C D2=A0000003 D3=434F4445
 =00000000 D5=00000005 D6=0000011C D7=00000000
0=00DFE1FF A1=000017D2 A2=00EFE1FE A3=0001A79A
4=0000CC28 A5=000F7634 A6=000F68BC A7=000F684E
DM 4185A8
185A8     0170 017C 2078 01E0  2478 01D4 2278 0134      .l.l x..$x.."x.4
```

$ 418564:

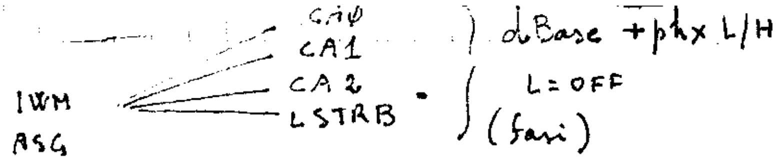
Set State control lines for Read/Write Registers conformemente a: D∅.B

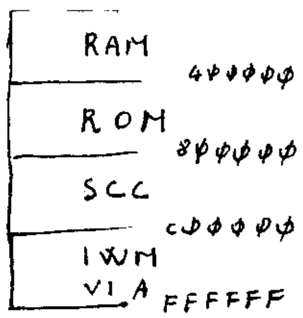


```
 DM 134
000134     0000 17D2 FFFF FFFF  FFFF 0000 0000 FFBF    ................
>  4105AC     05AC
4185AC:                      MOVEA.L $01E0,A0        IWM
4185B0:                      MOVEA.L $01D4,A2        VIA
4185B4:                      MOVEA.L $0134,A1   T_SonyPtr
4185B8:                      MOVE.W  $0012(A1),D1      (p.es. ∅∅∅6)
4185BC:                      ADD.W   D1,D1            (p.es. ∅∅∅C)
4185BE:                      MOVE.W  *-$0022(D1.W),D1 ; 0041859C +
4185C2:                      RTS                        p.e. 4185A8  : D1 ← ∅17C (p.es.)
4185C4:  set                 BSR.S   *-$0018          ; 004185AC
4185C6:                      TST.B   $0200(A0)        CA∅ = ON
4185CA:                      TST.B   $0600(A0)        CA1 = ON
4185CE:                      LSR.B   #$1,D0
4185D0:                      BCC.S   *+$0008          ; 004185D8
4185D2:                      TST.B   $0A00(A0)        CA2 = ON
4185D6:                      BRA.S   *+$0006          ; 004185DC
4185D8:                      TST.B   $0800(A0)        CA2 = OFF
4185DC:                      LSR.B   #$1,D0
4185DE:                      BCC.S   *+$000A          ; 004185E8
4185E0:                      BSET    #$05,$1E00(A2)     SEL = 1
4185E6:                      BRA.S   *+$0008          ; 004185EE
4185E8:                      BCLR    #$05,$1E00(A2)    SEL = ∅
>IL
4185EE:                      LSR.B   #$1,D0
4185F0:                      BCS.S   *+$0004          ; 004185F4
4185F2:                      TST.B   (A0)             CA∅ = OFF
   5F4:                      LSR.B   #$1,D0
4185F6:                      BCS.S   *+$0006          ; 004185FC
4185F8:                      TST.B   $0400(A0)        CA1 = OFF
4185FC:                      RTS
4185FE:  Read data bit       BSR.S   *-$003A          ; 004185C4   Set control lines
418600:                      MOVE    SR,-(A7)           Save Status
418602:                      ORI.W   #$0300,SR          Mask interrupts
418606:                      TST.B   $1A00(A0)          Q6 = ON (pass data from disk's RD/SENSE line)
41860A:                      MOVE.B  $1C00(A0),D0     read in D0 disk register's data bit (High bit)
41860E:                      TST.B   $1800(A0)          Q6 = OFF (reset IWM to idle)
418612:                      MOVE    (A7)+,SR          restore status
418614:                      TST.B   D0               ← Bit is ∅/1
418616:                      RTS
418618:  Write data bit      BSR.S   *-$0054          ; 004185C4   Set control lines
41861A:                      MOVE    SR,-(A7)
41861C:                      ORI.W   #$0300,SR
418620:                      TST.B   $0E00(A0)        LSTRB = ON
```

41864A −
418618
32

```
IL 418610 4186D2
418618:  Write data bit     BSR.S    *-$0054          ; 004185C4   Set control lines
41861A:                     MOVE     SR,-(A7)                      save status
41861C:                     ORI.W    #$0300,SR                     Mask interrupts
                                                                   Lstrb=ON
418620:                     TST.B    $0E00(A0)                     wait
418624:                     NOP                    }
418626:                     NOP
418628:                     TST.B    $0C00(A0)                     Lstrb=OFF
41862C:                     MOVE     (A7)+,SR                      restore status
                                                                   |1|1|ϕ|1|   CA1=ON    } ?     11ϕϕ
                                                                               CAϕ=ON
                                                                               SEL=ϕFF
41862E:                     RTS                                                CA2=ON
418630:  Drive Status?      MOVEQ    #$0D,D0
418632:                     BSR.S    *-$0034          ; 004185FE   Read data bit
418634:                     BPL.S    *+$0014          ; 00418648   bit was ϕ        single or double?
418636:                     TST.B    (A0)             ← bit was 1 : CAϕ=OFF         110ϕ
418638:                     BSR.S    *-$0038          ; 00418600   read bit
41863A:                     BPL.S    *+$000E          ; 00418648   single
41863C:                     TST.B    $0200(A0)                     CAϕ=ON  ??       1ϕ1ϕ
418640:                     TST.B    $0400(A0)                     CA1=OFF
418644:                     BSR.S    *-$0044          ; 00418600   read bit
418646:                     BRA.S    *+$0004          ; 0041864A              FFFFFFFF → Dϕ
418648:                     MOVEQ    #-$7F,D0
41864A:                     RTS
41864C:                     BSR      *-$0098          ; 004185B4
418650:                     MOVE.W   D6,D2
418652:                     LSR.W    #$4,D2
418654:                     LSL.W    #$3,D2
418656:                     ADD.W    D1,D2
418658:                     MOVE.W   $1A(A1,D2.W),D2
41865C:                     TST.B    $13(A1,D1.W)
418660:                     BMI.S    *+$0006          ; 00418666
418662:                     TST.W    D2
418664:                     BPL.S    *+$0004          ; 00418668
418666:                     MOVEQ    #$00,D2
418668:                     CMPI.W   #$01B8,D2
41866C:                     BLE.S    *+$0006          ; 00418672
41866E:                     MOVE.W   #$01B8,D2
418672:                     MOVE.W   $0138,D0
418676:                     BPL.S    *+$0008          ; 0041867E
418678:                     MOVE.W   $0038(A1),D0
41867C:                     BRA.S    *+$0018          ; 00418694
41867E:                     SUB.W    D2,D0
418680:                     BEQ      *-$00CC          ; 004185B4
418684:                     BPL.S    *+$0004          ; 00418688
418686:                     NEG.W    D0
418688:                     LSL.W    #$5,D0
41868A:                     CMP.W    $0036(A1),D0
41868E:                     BGT.S    *+$0006          ; 00418694
418690:                     MOVE.W   $0036(A1),D0
418694:                     ADD.W    $0014(A1),D0
418698:                     CMP.W    $0038(A1),D0
41869C:                     BLT.S    *+$0006          ; 004186A2
```





VIA ——— SEL : bit5 di vBase+vBufA

v Base sta in "VIA"(1D4) e normalmente va in A2
DCT (~~T~~-SonyVars) sta in "SonyVars" ($134) e normalmente va in A1

IWM:

dBase sta in "IWM"(global) e normalmente ~~va~~ in AØ
($1EØ)

CAØ
CA1
CA2
LSTRB } fasi

Enable : dBase + motor ON/Off

Select : dBase + ext/intDrive.

Q6 } ~~read~~ status register (read)
Q7 } data in (read)
data out (write) } select Q6/7 L/H

Read Disk registers 1) Q7 → OFF (R/W dBase + q7L)
2) Q6 = ON (R/W dBase + q6H)

Select register 3) abilitare il disco voluto (R/W dBase + int/ext Drive)
(+R/W dBase + Motor On)
4) Verificare LSTRB = Low (R/W dBase + ph3L
5) Selezionare il registro voluto:

~~CAØ~~
~~CA1~~
~~CA2~~
~~SEL~~

CA2	CA1	CAØ	SEL		
Ø	Ø	Ø	Ø	DIRTN	direzione testina
Ø	Ø	Ø	1	CSTIN	Disco inserito
Ø	Ø	1	Ø	STEP	Testina in movimento
Ø	Ø	1	1	WRTPRT	Protetto in scrittura.
Ø	1	Ø	Ø	MOTORON	Disco ~~in~~ rotazione
Ø	1	Ø	1	TKØ	Testina su Trk # Ø
Ø	1	1	Ø	?	
Ø	1	1	1	TACH	Tacheometro
1	Ø	Ø	Ø	RDDATAØ	Read data Side inf.
1	Ø	Ø	1	RDDATA1	Read data Side sup.
1	Ø	1	Ø	?	
1	Ø	1	1		
1	1	Ø	Ø	SIDES	Single/Double Side
1	1	Ø	1	?	
1	1	1	0		
1	1	1	1	DRVIN	Drive installato -

write disk register : 1) Accertarsi che LSTRB = off.

2) CAφ & CA1 → ON

3) Set SEL = φ

4)

CAφ	CA1	(SEL)		
φ	φ	(φ)	DirTn	Stepping Direction.
φ	1		Step	Step 1 traccia.
1	φ		Motoron	Disk Motor On/Off
1	1		Eject	Eject

Valore (bit!) da scrivere: su CA2

5) LSTRB → H (per ≥ 1µs, ≤ 1 ms) → L

NB: prima di cambiare SEL : CAφ, CA1 = H

EUA, an Extended User Area for TMON

by Darin Adler (December 2, 1986)

These are some ramblings about EUA, an Extended User Area for the TMON debugger.

Which Version?

The Extended User Area that this document describes is version 685. Source code is available and should be distributed with the EUA whenever possible (versions 667 and greater were created with the MPW assembler, available from APDA). EUA—including "discipline"—works well with the Macintosh and works reasonably well with the Macintosh Plus.

A quick way of determing the EUA version number from within TMON is entering `@user` in a NUMBER window and looking at the decimal number in parentheses. The high word of N is the size of the user area. If this value is $6000, then you have not used PatchTMON, or PatchTMON was unsuccessful (i.e. you ran it on a writeprotected disk, etc.). PatchTMON need not be used on TMON 2.600 or greater, including TMON20.

How to Install the Extended User Area

The extended user area is longer than TMON will normally allow. The user area is over $7000 bytes, long but standard TMON allows user areas of $6000 bytes or fewer. Included with the EUA is a program called PatchTMON. This will patch a copy of TMON, on the same disk, so that it will work with user areas of up to $7F00 bytes. This patch should not affect normal use of TMON, and I recommend patching all TMONs except the one on your original disk. To use PatchTMON, put a copy of it on the same disk (in the same folder) as TMON. Double-click PatchTMON. If an error occurs, PatchTMON will beep once.

Once the patch is applied, there are two ways to use the EUA with TMON. Both are documented in the TMON manual. The first way is to put the EUA on the disk with TMON and name it "User Area". When TMON installs itself it looks for a file called "User Area". If there is a file with this name on the same disk as TMON, it will load the file in instead of using the built-in user area inside TMON itself. The second way to load the EUA is to double-click the EUA icon from the Finder. This will install TMON and load and use the Extended User Area. (If you use this second method, the EUA can be given a file name different from "User Area".) Remember, however, that once TMON is installed with a particular user area, the machine must be shut down before it can be used with a different one.

Functions

EUA provides a number of new functions in the user area window. Some of these new functions are **Toggle screens**, **Template**, **Stack addresses**, **Stack crawl**, **Look for labels between LINK/UNLK of Ax**, **Click mouse outside TMON**, **Windows**, **Launch**, **Shut down** and **Trap discipline**. The user area window has been divided into three pages, so that the window will take up less room on the screen. The EUA also provides a built-in table with labels for all the low memory globals described in the *May 1985 Software Supplement* plus many others.

The **Toggle screens** function is used to choose between the three pages of user area functions. Click on this line and press Return or Enter to cycle through the pages.

The **Template** function displays memory dumps in a nice format for a few data structures. Press Return on the line until the structure that you want appears between curly brackets (either **WindowRecord**, **ControlRecord**, **TERec** or **ParamBlock**). Then, type the address of a structure that you wish to examine. To follow a window or control list, press Enter.

The **Stack addresses** function is useful for searching the stack for a valid address. Press Enter on the line and valid addresses will be presented, one by one.

The **Stack crawl** function is useful for examining the call chain of routines that start with `LINK A6` and end with UNLK A6. By pressing Enter repeatedly on the line, the call chain will be traversed.

Look for labels between LINK/UNLK of Ax is a function which causes the EUA to look for labels between LINK and UNLK statements of registers other than A6. To switch to a different address register, type the number of the register on this line. To use TMON with some FORTH interpreters, it may be useful to switch to register A2.

Click mouse outside TMON is a new name for the **ShowScrn** function from the original user area.

Windows is a function that sets up a default arrangment windows that is used the Monitor starts up. Set up the windows the way that you like and then type a number on the line. 0 means that you want the windows configured just as they are when you choose the function; 1 means that you do not want the `USER` window to be included. These windows will be saved when the user area is saved using TMON's configuration button.

Launch allows you to exit to the Finder or re-launch the application (for another try at debugging). Either function closes all open files before launching. 0 launches the Finder; 1 launches whatever is stored in the low-memory global `CurApName`.

Shut down shuts down the machine. 0 resets; 1 unmounts and ejects all volumes before resetting.

Trap discipline is a parameter-checking front end to the Mac ROM. It is exactly like **Trap intercept**, except that a trap is only intercepted if one of its parameters is invalid. There is a strict and a lenient version of this discipline. The strict version is recommended for writing new programs, while the lenient can be left on all the time, to find out about bugs in existing programs. To toggle between the two different levels, press Return alone on this line.

New ROM Features and Bugs

The EUA provides a number of features that make TMON better for use with the 128K ROMs.

The trap table built into TMON is bypassed and a new one, built into the EUA is used. All of the names for new traps that Apple has documented are provided in the user area (an internal trap name table is used, and built in trap names of TMON are turned off).

Trap numbering has been changed. With the old ROMs there were 512 traps, numbered $0 to $1FF. With the new ROMs (and whenever you use this user area), there are 768 traps, numbered $0 to $FF (OS traps) and $800 to $9FF (ToolBox traps). Because of this new trap numbering, all the trap numbers typed into the **Trap record**, **Trap (heap check)**, **Trap discipline**, **Trap checksum**, **Trap intercept**, and **Trap signal** functions are somewhat different. Previously, typing `0 1FF` was a way to select all traps. With the new user area (old or new ROMs), `0 9FF` or `0 FFF` is the correct way; typing `0 1FF` will select only OS traps ($0 to $FF).

There are a number of new traps available from the new ROMs. In addition, many traps are actually vectors used by the ROMs: useful hooks for changing the behavior of the Mac operating system. Apple documentation has not described what these are, but Apple "private" equate files have given them names.

Patches in EUA

The extended user area performs several patches on TMON for compatibility of TMON with other products (usually due to bugs in the other products, not in TMON).

One patch fixes an incompatibility with TMON and GCC's HyperDrive 20. GCC's newer software releases are modified so that this problem is corrected both by TMON and the HyperDrive software.

Another patch makes the `_Debugger` trap work properly and handles unimplemented A000 traps properly. There is a patch for something bad the AppleTalk driver interface code does (it checks the value at address $E0, which is *reserved* by Motorola and *changed* by TMON).

There is a patch so that TMON will work with free form sound. This patch was necessary because of some buggy code in some old versions of the System.

TMON was patched to call the extended user area so that the background screen is the application screen and not just the grey TMON background. In addition, when using EUA, TMON does not restore the mouse coordinates from the application. This used to cause the mouse to jump around after a STEP, TRACE, GOSUB, or EXIT.

Another patch causes TMON to recognize embedded procedure names in resources other than 'CODE' resources (for example, 'WDEF' and 'CDEF' resources). A complete list of which resource types work with embedded procedure names is in the source code.

Yet another patch causes TMON to identify addresses pointed to by the dispatch table in all heap blocks. Previously, the identification was only done in non-relocatable blocks.

A patch keeps TMON from misbehaving when an empty resource file is open.

The EUA changes TMON's behavior with the *_SysError* trap. Standard TMON will intercept all SysError traps except for those with DS error codes of 30 and 31 (disk swapping "dialogs"). Some applications may use the additional DS codes for "dialogs" of their own. The EUA will only trap codes that are used for errors (codes between $01 and $1E).

New Stuff

A number of things have improved in the EUA since version 665. For those who have been using 665, here is a partial list of changes.

Discipline has been improved again, please send me any suggestions you might have about future improvements.

The launch function now resets the stack pointer to `CurStackBase`, this makes it work, even after a stack sniffer error.

This version of the EUA works with the TMON 68020 interim version, TMON20, available from ICOM, and will work in this way on some 68020 Macs. In addition, this version should be compatible with most large screens, if you use it with a new TMON (version 2.614 or greater).

I have added ability to look for labels between LINK/UNLK instructions of registers other than A6. This should be useful to some folks who use TMON with FORTH interpreters.

The heap scramble function has been improved. If you have had trouble with it in the past, you may want to try using heap scramble again. It now scrambles in fewer cases, and should no longer disturb properly written programs.

A bug that was present in version between 667 and 683 was fixed. This means that the **Windows** function should work properly again.

Wish List

A few items that I might want to address in the future:

The goal for lenient discipline is to establish a flavor of discipline that only reveals bugs, rather that strange idiosyncrasies of working programs. While the original, strict discipline remains useful (especially for working on new programs), the new lenient discipline may be suitable to leave on most of the time. Currently, the only difference between lenient and strict discipline is the handling of rectangles, `SetTrapAddress`, and the result of `GetScrap`. In the future, lenient discipline will only complain about illegal parameters, not questionable ones.

Discipline is going to be extended to handle a variety of low-level routines, for example `jIODone`, as well as A000 traps.

Many functions of the Extended User Area will be standard in a future debugging product.

Optional Patch

TMON was designed to work with the Mac, but it will work well with the Mac Plus. Unfortunately, the low-memory global that TMON uses to store its status information ($120, otherwise known as `MacJmp`), is used by the new ROMs for debugger support. Because of this, TMON will work on the Plus, but when you open the TMON icon a second time, TMON will not know that it is already installed. A workaround is to patch TMON to use the location $AE0 instead (this is unused by the system, so far). Simply search for the bytes 00000120 in TMON and replace them with 00000AE0. This will not cause any problems with the 512, and will make TMON work better with the Plus.

Other Information

EUA is *not* a product of ICOM Simulations (distributors of TMON). It is free of charge. The debugger, TMON, is available from ICOM Simulations, (312) 520-4440. The latest version of the EUA is usually available from ICOM. The author of EUA can be contacted by U.S. Mail at 2765 Marl Oak Dr., Highland Park, IL 60035, or on

GEnie (and sometimes Delphi) with usemame DADLER.

Issue 1

TMON News

This is the first TMON newsletter. The purpose of this newsletter is to inform TMON users of news regarding the product and to provide answers to frequently asked questions.

WHO IS ICOM SIMULATIONS, INC.?
Many TMON users only know of TMQ software, Inc. as the people behind TMON. ICOM Simulations is the new distributor and supporter of TMON. In fact, for the most part we are the same people. ICOM Simulations is our new name. We also have a new address which is on the bottom of this newsletter. Please use this address for all correspondence dealing with TMON.

WHAT VERSION ARE WE AT?
Just to confirm another frequently asked question – the current version of TMON is 2.585. There has been only one modification recently, and that was only to change some text from TMQ Software to ICOM Simulations, Inc. on the menu screen. This by itself did not justify changing the version number.

WHAT EXTENDED USER AREA?
Some of you might already have come across a PUBLIC DOMAIN Extended User Area (EUA) on CompuServe, Delphi, or elsewhere. To summarize some of its features, 1) the background screen is the application screen, NOT the grey TMON background screen, 2) mouse coordinates are not restored from the application with each Step/Trace/Gosub, ... so the mouse moves around the way you would expect it to, 3) there is a stack crawl, 4) there is a built in discipline to check parameters to traps, ... and other nice extensions as well. This is a public domain user area (with source code), and as such is not supported by ICOM Simulations. However, many may prefer this debugging environment over the ones currently supplied with TMON. The latest version of EUA seen at the time of this writing is version 685. (Special thanks to Darin Adler for putting it together; Bill Steinberg for putting it on CompuServe; and Peter Olson for putting it on Delphi.) We encourage other user areas to be distributed as public domain to support your fellow TMON users.

_Debugger ALTERNATIVES.
Many people have found that _Debugger does not work in TMON as it does in MacsBug. The easiest alternative is to use the 68000 instruction TRAP #n, where 'n' is 0..15. (TMON uses TRAP #$F for breakpoints (see p. 22 of the manual), but unless you set a breakpoint so that it is in TMON's table of breakpoints, TRAP #$F will be recognized as a TRAP and the program will fall into TMON.) _Debugger DOES work with the Extended User Area mentioned above.

REFRESHING VECTORS WHILE DEBUGGING EXISTING APPLICATIONS.
Some applications (i.e. MacWrite!) redirect vectors and EXPECTS them redirected in order to run. TMON can be configured to automatically refresh these vectors in the configuration section. This can sometimes be a nice thing, but sometimes it can cause problems. The solution – sometimes you MUST disable the Vector Refresh option when debugging existing applications. (See p. 33 of the manual "Debugging Existing Applications" for more information.)

About EUA (Extended User Area)

June 1987

As many TMON users already know, TMON is an extensible debugger. One of TMON's most popular extensions is a public domain user area written by Darin Adler which has become known as EUA (Extended User Area). EUA is not a product of ICOM Simulations. Several versions of EUA have been available on CompuServe and Delphi. However, due to the popularity of EUA, ICOM Simulations is now providing an EUA disk with TMON.

EUA provides functions which extend the default user area provided with TMON and adds some features which take advantage of the new 128K ROMS. In addition, some compatibility problems with TMON and the 128K ROMS are resolved by EUA. Complete source code and a documentation file to EUA are also included on the EUA disk.

OTHER PRODUCTS BY *ICOM Simulations*:

Déjà Vu: A Nightmare Comes True

Winner of Software Publishers' Association's Best Entertainment Product and Best New World awards for 1985, and winner of MacUser Magazine's Best Entertainment Program for 1985.

This is a state-of-the-art interactive adventure game for the Macintosh developed by ICOM Simulations, Inc. TMON was the debugger used during its development. *Déjà Vu* is a mouse driven game which makes full use of the Mac user interface.

Uninvited

This is another interactive adventure using the same award winning game system as *Déjà Vu: A Nightmare Comes True.*

It is the winner of numerous awards, including: MacUser Magazine's Entertainment Product of the Year, 1986; Omni Magazine's Best Macintosh Game, 1986; and nominations for the 1986 Software Publishers' Association's "Excellence in Software" awards in four categories: Best Entertainment, Best Graphics, Best Concept, and Best User Interface.

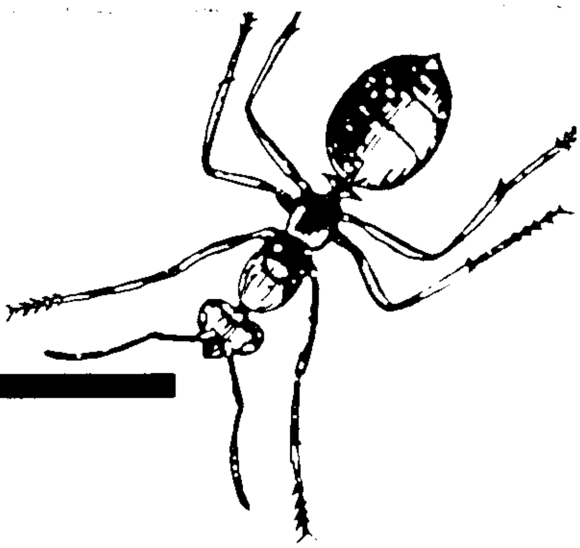
Shadowgate

The latest in the series of interactive adventure games, developed by ICOM Simulations, Inc., and using the same award winning game system as *Déjà Vu: A Nightmare Comes True* and *Uninvited.*

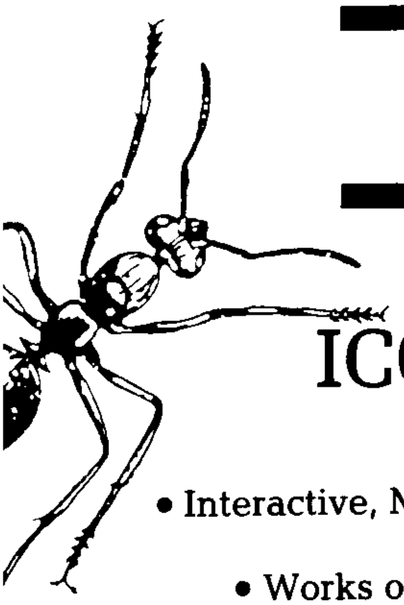
> *Uninvited, Déjà Vu: A Nightmare Comes True,* and *Shadowgate* are being marketed by Mindscape Inc. To find out where to purchase them in your area, call Mindscape customer service at 1-800-221-9884 (in Illinois, 1-800-942-7315).

Future products?

Keep your eyes open for other quality software from ICOM Simulations, Inc.



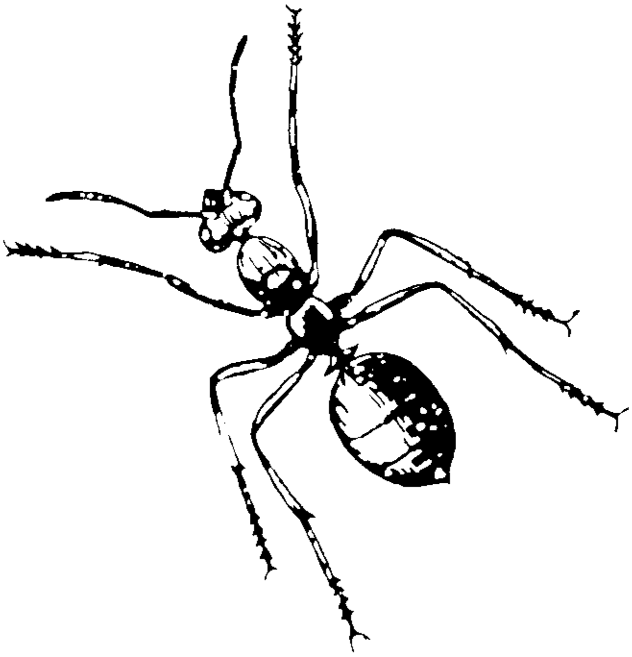
TMON



ICOM Simulations Inc.

formerly TMQ Software, Inc.

- Interactive, Multi-window Monitor/Debugger for the Macintosh™
- Works on Both 128K and 512K Versions of the Machine
- An absolute must for any serious development on the Macintosh.™



Programmed by Waldemar Horwat exclusively for TMQ Software, Inc.